Entwicklung und Aufbau eines Überwachungs- und Kontrollsystems für das Crystal-Barrel-Experiment an ELSA

von

Torge Szczepanek

Diplomarbeit in Physik

angefertigt im
Institut für Strahlen- und Kernphysik

vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
Bonn

im April 2001

Ich versichere, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Referent:	Prof. Dr. E. Klempt
Korreferent:	Priv. Doz. Dr. R. W. Gothe

Inhaltsverzeichnis

Kapitel 1

Einleitung

Moderne Kern- und Teilchenphysikexperimente sind mittlerweile so komplex geworden, dass sie eine zentrale Steuerung und Überwachung (slow control) der einzelnen Detektorkomponenten benötigen. Diese Diplomarbeit beschreibt die Entwicklung und den Aufbau eines solchen Steuerungs- und Überwachungssystems für das Crystal-Barrel-Experiment, wie es an der Elektronen-Stretcher-Anlage ELSA in Bonn aufgebaut ist.

Nach langjährigem erfolgreichen Einsatz am Low-Energy-Antiproton-Ring am CERN wurde der Crystal-Barrel-Detektor in Bonn aufgebaut. Ziel der Untersuchungen ist ein besseres Verständis der Quantenchromodynamik im mittleren Energiebereich. Hierzu werden photoinduzierte Nukleonresonanzen untersucht. Die ersten experimentellen Zielsetzungen und ihre Einbettung in breitere experimentelle Fragestellungen sind in einer Reihe von Experimentiervorschlägen beschrieben (siehe [1], [2] und [3]). Die Vorschläge wurden von der CB-ELSA-Kollaboration[1] erarbeitet und vom Program Advisory Committee (PAC) zur Durchführung empfohlen. Das PAC begleitet das experimentelle Programm an ELSA und dem Elektronenbeschleuniger MAMI in Mainz

Die jetzige Zusammensetzung des Detektors erforderte die Entwicklung eines neuen Slowcontrol-Systems, das die neuen Detektorkomponenten berücksichtigt und das alte Slowcontrol-System vom CERN ersetzt.

Zunächst folgt nun eine Übersicht über das gesamte Experiment, wie es zum gegenwärtigen Zeitpunkt an der Elektronen-Stretcher-Anlage installiert ist; die einzelnen Detektorkomponenten, die zu überwachen bzw. zu steuern sind, werden hier vorgestellt.

Die Arbeit verfolgt eine doppelte Zielsetzung: zum einen soll das im Laufe der vergangenen Monate aufgebaute Überwachungssystem beschrieben und dokumentiert werden. Zugleich ist diese Arbeit aber auch gedacht als Handbuch oder Bedienungsanleitung für die Benutzung und Weiterentwicklung des Systems. Diese zweite Funktion erfordert die Verwendung vieler Fachbegriffe, die das Lesen der Arbeit zuweilen erschweren mögen. Für denjenigen, der mit dem System tatsächlich arbeiten möchte, ist es jedoch als Erleichterung und Hilfe gedacht.

[1] Crystal-Barrel an ELSA Kollaboration

Kapitel 2

Aufbau des CB-ELSA-Experiments

Nachdem durch den Abbau des SAPHIR-Experiments [4] ein Strahlplatz frei geworden war, wurde im Jahr 1999 in Bonn der Crystal-Barrel-Detektor an der Elektronen-Stretcher-Anlage ELSA aufgebaut. Hier soll zunächst ein kurzer Überblick über die Elektronen-Stretcher-Anlage gegeben werden. Danach wird dann der Detektor mit allen Komponenten beschrieben.

2.1 Elektronen-Stretcher-Anlage

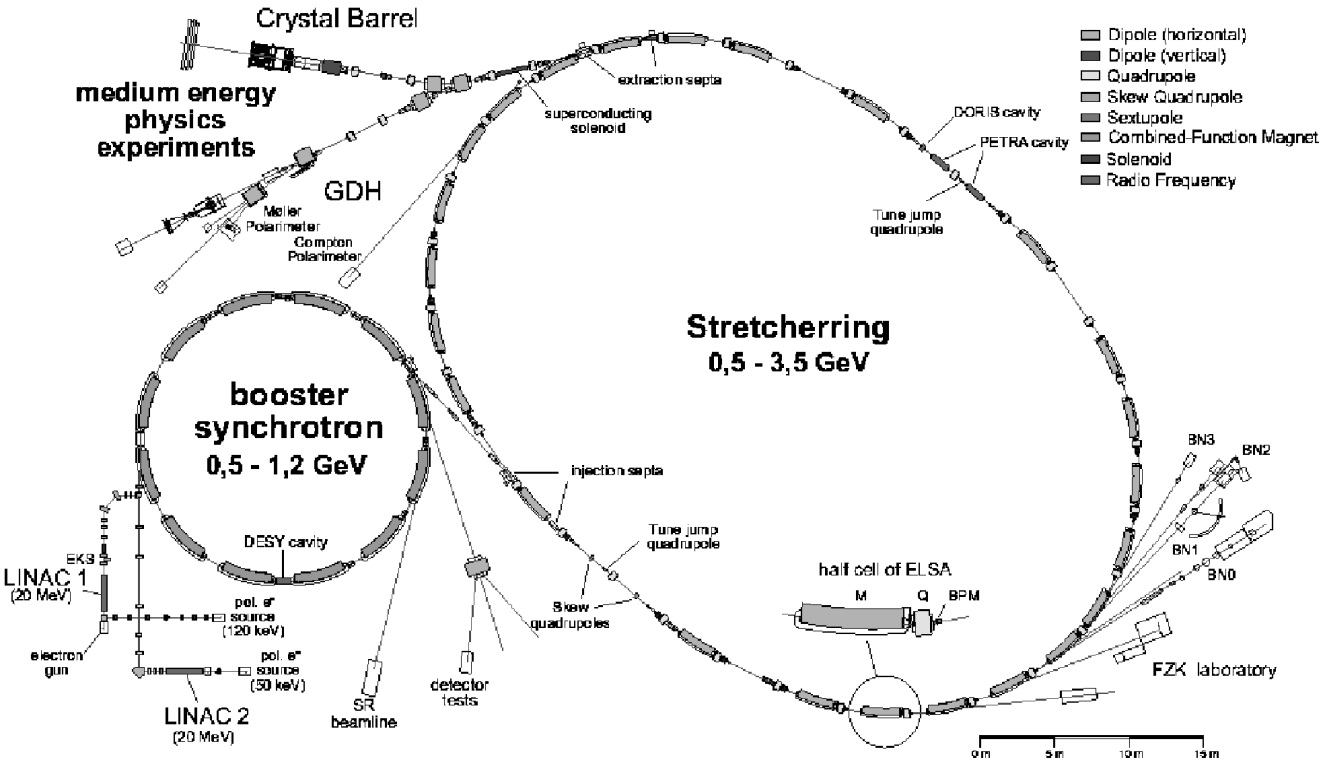

Abbildung 2.1: Beschleunigeranlage ELSA

Abbildung 2.1 enthält einen schematischen Überblick über die gesamte Anlage. Die von einer Gun erzeugten Elektronen werden mit Hilfe eines **LINAC**s[1] auf eine Energie von ca. 20 MeV beschleunigt und in das Booster-Synchrotron eingespeist. Dort können die Elektronen auf eine Energie von bis zu 1.2 GeV weiterbeschleunigt werden. Die nachgeschaltete Elektronen-Stretcher-Anlage akkumuliert den vom Synchrotron erzeugten Elektronenstrahl und verteilt die Elektronenpakete[2] auf den gesamten ELSA-Ring (stretching), so dass eine fast kontinuierliche Füllung im Ring erreicht wird. ELSA kann nun in zwei verschiedenen Modi betrieben werden:

- **Speichermode:** die Elektronen verbleiben im Ring. Es wird ein hoher Strom akkumuliert
- **Boostermode:** die Elektronen im Ring werden weiter beschleunigt (bis auf maximal 3.5 GeV) und dann zum jeweiligen Experiment extrahiert

Der erste Modus wird verwendet, um Synchrotronlicht zu erzeugen, das von Elektronen produziert wird, die durch Magnete auf eine gekrümmte Bahn gelenkt werden. Dieses stark fokussierte Licht wird für Synchrotronlichtexperimente verwendet. Der zweite Modus ist der für das Crystal-Barrel-Experiment interessante Betriebsmodus. Durch die gleichmäßige Verteilung der Elektronen im Ring kann ein Tastverhältnis (Verhältnis von Extraktionszeit zu Zykluszeit) von nahezu 100% erreicht werden, so dass ein relativ gleichmäßiger Elektronenstrom extrahiert werden kann. Die Elektronen-Stretcher-Anlage verfügt über ein komplexes Steuerungs- und Überwachungssystem, welches über mehrere Jahre entwickelt wurde. Die vorliegende Arbeit befasst sich ausschließlich mit der Steuerung und Überwachung des Crystal-Barrel-Experiments.

2.2 Erzeugung von γ-Quanten und Taggingsystem

Um photoinduzierte Reaktionen untersuchen zu können, muss aus dem primären Elektronenstrahl zunächst ein Photonenstrahl gewonnen werden. Dies geschieht mit Hilfe eines Radiatortargets, an dem Bremsstrahlungsphotonen erzeugt werden. Ein solches Bremsstrahlungsphoton fliegt in der ursprünglichen Strahlrichtung weiter, während das Elektron Energie verliert und im Magnetfeld des Taggingmagneten nach unten abgelenkt wird. Je größer der Energieverlust des Elektrons, desto stärker wird es im Feld abgelenkt. Bestimmt man nun die Energie dieses Elektrons, so kann man mit Kenntnis der ursprünglichen Energie E_0 im ELSA-Ring die Energie des auslaufenden Photons E_γ ermitteln:

$$E_\gamma = E_0 - E_e$$

Die Energie des Elektrons E_e kann mit Kenntnis der Magnetfeldstärke B des Taggingmagneten bestimmt werden. Für $pc \gg m_e c^2$ gilt:

$$E_e = pc = eBrc$$

[1] Linear Accelerator
[2] Bunches

Wichtig für die Überwachung des Experiments ist somit das Magnetfeld des Taggingmagneten. Es wird fest eingestellt und sollte sich - um reproduzierbare Ergebnisse zu erhalten - nicht ändern.

Die Energiebestimmung der abgelenkten Elektronen erfolgt mit dem Taggingsystem des SAPHIR-Experiments [4]. Dieses besteht aus 14 Szintillatoren [5] und zwei Drahtkammern [6] mit insgesamt 352 Drähten. Das Taggingsystem wurde hierzu für die Verwendung am Crystal-Barrel-Experiment modifiziert.

Das Energiespektrum der Bremsstrahlungselektronen folgt einem $1/e$-Verlauf. Der oberste Szintillator erfährt somit die höchste Rate. Aus diesem Grund sind die oberen Zähler schmal, nach unten nimmt die Breite der Szintillatoren zu. Das System deckt einen Energiebereich von 22% bis 93% von E_0 ab. Die Proportionaldrahtkammern haben eine theoretische Energieauflösung von 0.028% bis 2.6%. Die Szintillatoren des Taggingssystems werden mit Photomultipliern[3] ausgelesen, die an eine Hochspannungsquelle angeschlossen sind. Jeder Szintillator wird von beiden Seiten mit jeweils einem Photomultiplier ausgelesen, so dass insgesamt 28 Hochspannungen benötigt werden. Bei zu hohen Raten in den einzelnen Szintillatoren kann es vorkommen, dass der Strom zu groß wird und die Hochspannungsversorgung den jeweiligen Kanal abschaltet, so dass eine sinnvolle Datenerfassung mit dem jeweiligen Szintillator nicht mehr möglich ist.

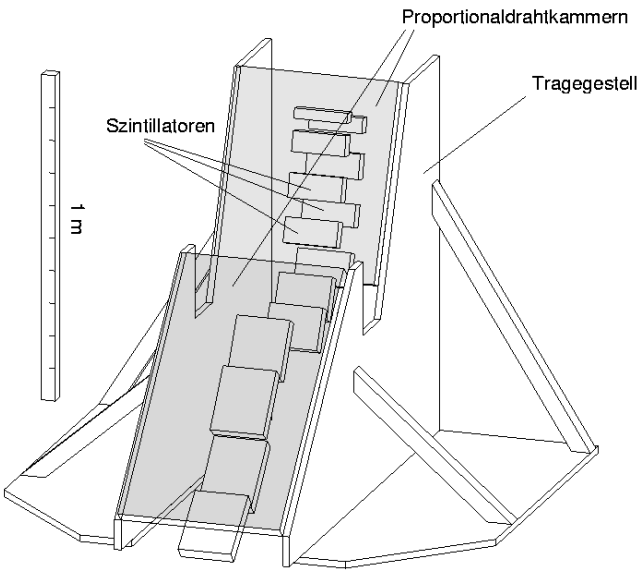

Abbildung 2.2: Taggingsystem TOPAS II

Die Drahtkammern werden mit einer Spannung von ca. 4 kV bis 4.5 kV betrieben. Im normalen Betrieb wird ein Stromfluss von einigen μA erreicht. Wird der Elektronenfluss zu groß, so kommt es zu Überschlägen, die die Drahtkammern beschädigen können. Eine Überwachung und Steuerung der Spannungen und vor allem die Überwachung des Stromflusses wird somit zum Schutz der Kammern dringend benötigt.

[3] Sekundärelektronenvervielfachern

2.3 Das Flüssigwasserstoff-Target

Die vom Radiator erzeugten Bremsstrahlungsphotonen treffen im Zentrum des Crystal-Barrel-Kalorimeters auf eine Targetzelle, die mit flüssigem Wasserstoff gefüllt ist und somit Protonen für die Resonanzreaktionen zur Verfügung stellt. Die Targetzelle besteht aus einem 5 cm langen Zylinder aus Kaptonfolie mit einem Durchmesser von 3 cm. Das Targetsystem besteht aus zwei getrennten Kreisläufen, die über einen Wärmetauscher miteinander verbunden sind.

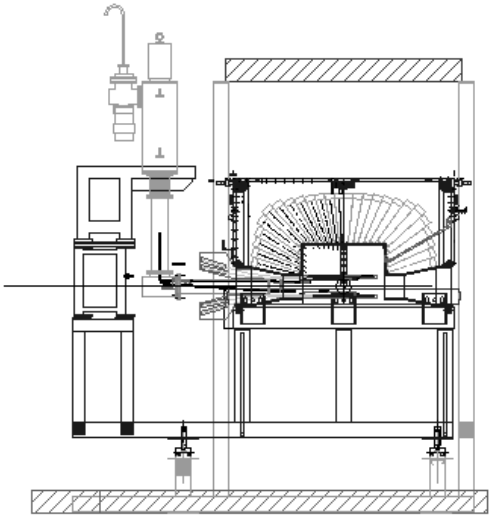

Abbildung 2.3: Targetsystem und Crystal-Barrel-Kalorimeter

Im Sekundärkreislauf wird ein Reservoir verwendet, dessen Pegelstand mit Hilfe einer Leveldiode ausgelesen wird. Die Verflüssigung des Wasserstoffs erfolgt im Primärkreislauf, der die Targetzelle enthält. Der Druck der Targetzelle wird mit einem Drucksensor bestimmt. Das Flüssigwasserstoff-Target wurde auf der Grundlage des Targetsystems des CB-LEAR Experiments aufgebaut [7].

2.4 Der Innendetektor

Um geladene Teilchen aus der Zerfallsregion und insbesondere die Protonen zu detektieren, wurde für das CB-ELSA-Experiment ein komplett neuer Innendetektor entworfen [8]. Dieser besteht aus insgesamt 513 szintillierenden Fasern in drei Lagen. Die Fasern in der äußersten Lage sind parallel zur Strahlachse angebracht. In den anderen beiden Lagen verlaufen die Fasern unter einem Winkel von $+25°$ bzw. $-25°$ dazu. Dies ermöglicht eine Rekonstruktion von Durchstoßpunkten geladenener Teilchen. Die Fasern werden mit Vielfach-Photomultipliern ausgelesen, die ihre Signale über Lichtleiterkabel zugeführt bekommen.

Die Wechselwirkungswahrscheinlichkeit der Photonen und anderer neutraler Teilchen (z.B. Neutronen) mit den Fasern ist sehr gering, da die Dicke des Detektors wesentlich kleiner als die Strahlungslänge ist, so dass nahezu kein Signal erzeugt wird.

2.5 Das Crystal-Barrel-Kalorimeter

Das Kalorimeter ist ein fassförmiger Detektor mit insgesamt 1380 CsI(Tl)-Kristallen, die nahezu den gesamten Raumwinkelbereich von 4π abdecken (97.8%). Die Kristalle sind in insgesamt 26 Ringen um das Targetsystem herum angeordnet. In den drei jeweils äußeren Ringen befinden sich 30 Kristalle. Die 20 mittleren Ringe enthalten 60 Kristalle. Jeder einzelne Kristall deckt einen Winkelbereich von 12° in ϕ- und 6° in θ-Richtung ab. Insgesamt wird ein Azimutalwinkelbereich von 12° bis 168° abgedeckt. Der Detektor und seine Auslese werden in [9] beschrieben. Die Auslese wurde - den Bedürfnissen des Experiments entsprechend - neu aufgebaut [10].

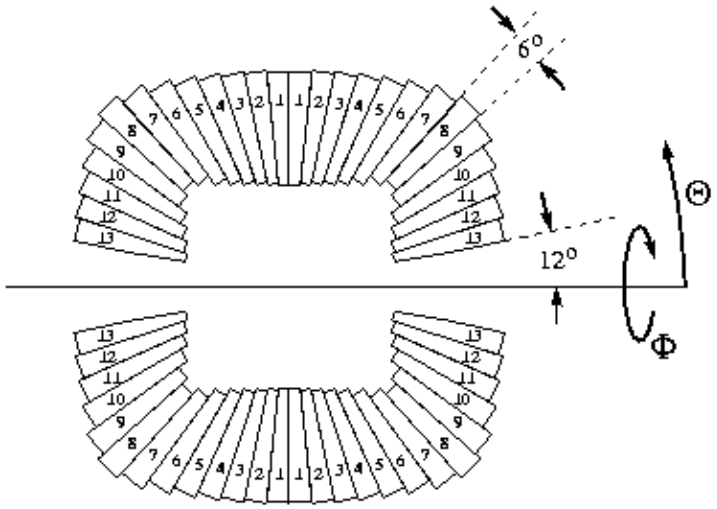

Abbildung 2.4: Die Kristalle des Crystal-Barrel-Kalorimeters

Das Kalorimeter ist sehr gut geeignet, Photonen zu detektieren, die zum Beispiel als Folgeprodukte von neutralen Mesonen (z.B. π_0 und η) entstehen. Die Kristalle haben für Photonen eine Strahlungslänge von $L_R = 1.86$ cm, so dass bei einer Länge von 30 cm (16.1 L_R) ein sehr großer Teil der Schauerenergie im Kristall deponiert wird. Das mit Photodioden gesammelte Licht ist direkt proportional zur deponierten Energie. Durch die Photodiode wird das Licht in einen Photostrom gewandelt, der mit einem ADC[4] gemessen werden kann.

Da die Lichtausbeute der Kristalle temperaturabhängig ist, gibt es in den einzelnen Crystal-Barrel-Segmenten Temperatursensoren, die jeweils die Temperatur für einen Sektor bestimmen. Der gesamte Detektor muss über eine Kühlanlage auf konstanter Temperatur gehalten werden.

Die gesamte Ausleseelektronik für das Kalorimeter ist separat in der PHOENICS-Halle untergebracht, die etwa 25 m entfernt liegt und während des Betriebs des Experiments zugänglich ist. Zur Ausleseelektronik gehören Shaper, Diskriminatoren, ADCs und die lokalen Eventbuilder.

[4] Analog to Digital Converter

2.6 Vorwärtsdetektoren

2.6.1 Das Flugzeitspektrometer

Das Flugzeitspektrometer besteht aus vier Flugzeitwänden, die aus jeweils 14 Szintillatoren zusammengesetzt sind und für das ELAN-Experiment entwickelt und gebaut [11] wurden. Die Zähler haben eine Länge von 3 m, eine Breite von 20 cm und eine Dicke von 5 cm. Sie werden beidseitig mit Photomultipliern ausgelesen. Insgesamt werden $4{\cdot}14{\cdot}2 = 112$ Hochspannungen[5] für die Photomultiplier benötigt. Über die Signallaufzeiten in den Szintillatoren kann der Durchstoßpunkt eines Teilchens genauer als 5 cm bestimmt werden. Die Signallaufzeiten werden mit TDCs[6] bestimmt.

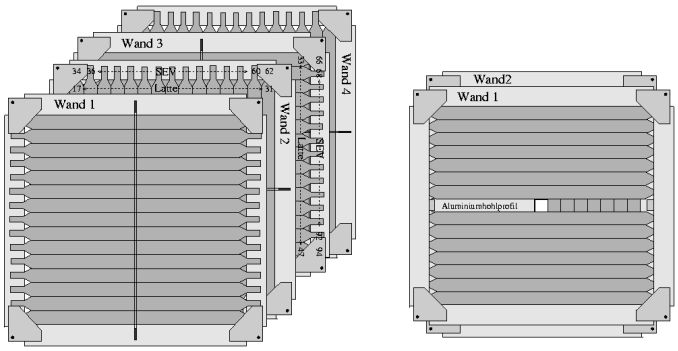

Abbildung 2.5: Das Time-of-flight System

Die Latten von Wand 1 und Wand 3 sind horizontal angebracht, die Latten von Wand 2 und Wand 4 vertikal. Somit kann der Durchgangspunkt eines Teilchens auf $10 \cdot 10\ cm^2$ genau bestimmt werden. Die TOF-Wände decken genau den Öffnungswinkel des Barrel-Kalorimeters ab.

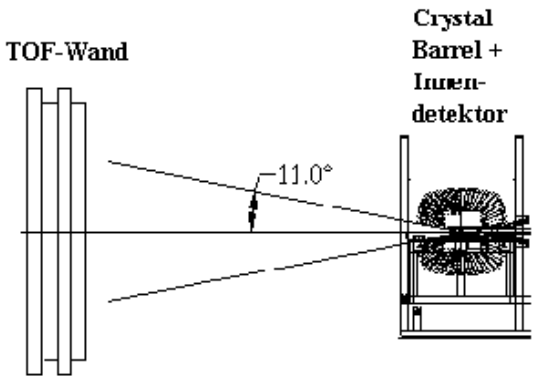

Abbildung 2.6: Öffnungswinkel des Barrel-Kalorimeters und TOF-Wände

Die Detektion von Teilchen in Vorwärtsrichtung ist bei der Photoproduktion von Mesonen nahe an der Schwelle wichtig, da dabei Rückstoßprotonen unter kleinen Winkeln emittiert werden. Weitere Informationen über das Flugzeitspektrometer finden sich in [12].

[5] 4 Wände, 14 Szintillatoren je Wand, 2 Photomultiplier je Szintillator
[6] Time to digital converter

2.6.2 Der TAPS-Detektor

Ab Mitte 2001 wird der TAPS[7]-Detektor als weiterer Vorwärtsdetektor zwischen dem Barrel und den TOF-Flugzeitwänden aufgebaut werden, nachdem dieser am A2-Experiment am Mainzer Microtron (MAMI) abgebaut wurde. Der TAPS-Detektor [13] besteht aus BaF_2-Modulen, die in verschiedenen Aufbauten eingesetzt werden können. Für den Einsatz am CB-ELSA-Experiment ist eine hexagonale Struktur mit 522 Modulen vorgesehen. Die Öffnung von 12° des Crystal-Barrel-Kalorimeters, die dem TAPS-Detektor zugewandt ist, wird dann durch Entfernen einzelner Kristallringe auf 30° erhöht. Die Verwendung von schnellen BaF_2-Szintillatoren ermöglicht die Verwendung von TAPS als schnellem Vorwärtsdetektor, der eine hohe Zählratenfestigkeit besitzt.

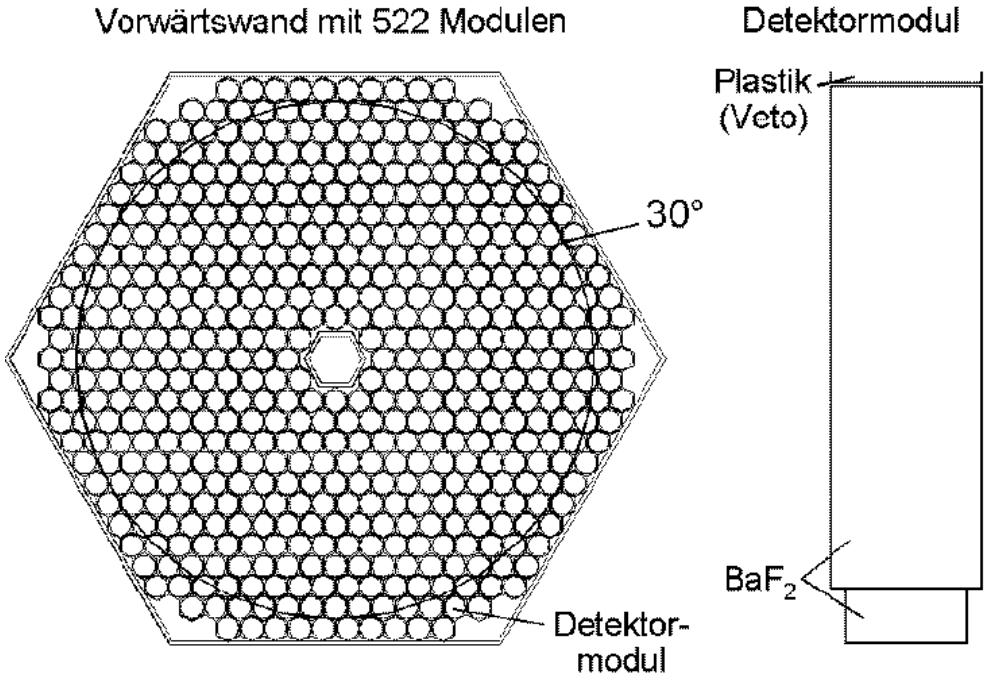

Abbildung 2.7: Der Taps-Detektor

2.6.3 Weitere Vorwärtsdetektoren

Baryonresonanzen können auch durch Elektroproduktion (mithilfe virtueller Photonen) erzeugt werden. Hierzu müssen die gestreuten Photonen nachgewiesen und ihre Energie und Flugrichtung gemessen werden. Dies kann in einer Bleiglaswand [14] oder in einem elektromagnetisches Spektrometer (EMS) [15] erfolgen. Das EMS ist, genau wie die TOF-Wände, für die Untersuchung der Mesonenproduktion (z.B. von η, ω und η') direkt an der Schwelle geeignet, wobei die Reaktionsprotonen unter kleinen Winkeln mit hoher Präzision vermessen werden können.

[7] Two Arm Photon Spectrometer oder auch Travel Around Photon Spectrometer

2.7 Der Gamma-Veto-Detektor

Vom Radiator erzeugte Photonen, die kein Ereignis im Flüssigwasserstoff-Target auslösen und dieses unbeeinflusst passieren, werden mit dem Gamma-Veto-Detektor nachgewiesen. Wird ein Photon im Gamma-Veto-Detektor registriert, so wird ein Veto-Signal an die zentrale Triggerlogik übergeben, dass kein physikalisch interessantes Ereignis aufgetreten ist. Die zentrale Triggerlogik kann somit dieses Ereignis verwerfen.

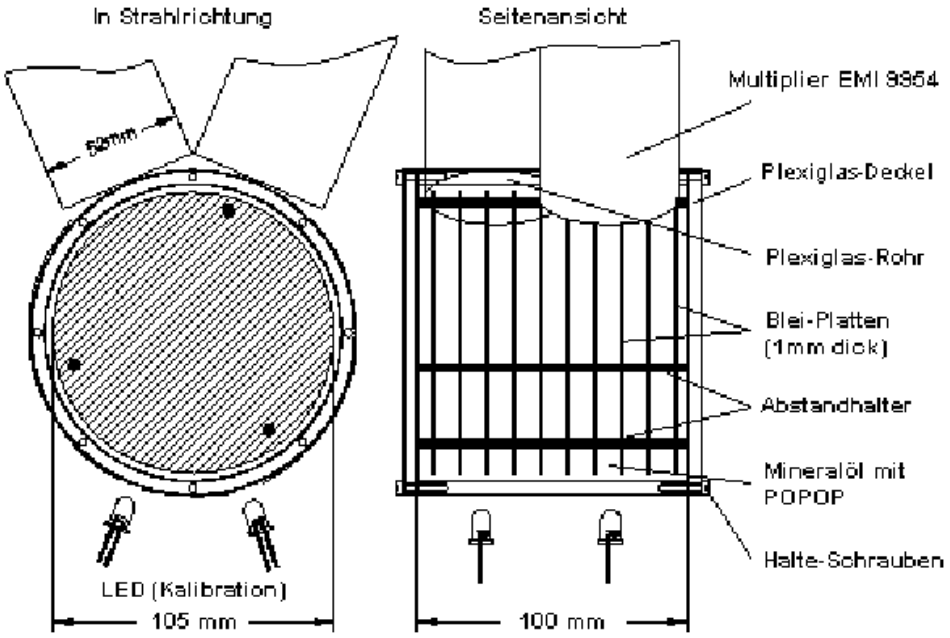

Abbildung 2.8: Der Gamma-Veto-Detektor

Der Detektor ist ein Cherenkov-Detektor: Ein eintreffendes Photon löst einen elektromagnetischen Schauer aus, dessen Partikel im Mineralöl Cherenkov-Licht abgeben, welches von Photomultipliern nachgewiesen wird.

Der Detektor besteht aus insgesamt drei zylinderförmigen Modulen mit 15 cm Durchmesser und 10 cm Länge, die in Sandwichtechnik aus 1 cm dicken Bleischeiben[8] und Mineralöl aufgebaut sind. Die Bleischeiben sind im Abstand von 10 mm angebracht. Die Zwischenräume sind mit Mineralöl gefüllt. Darin befindet sich ein Wellenlängenschieber (POPOP), der das gerichtete Cherenkov-Licht im Bereich von ca. 310-380 nm in isotropes Licht im 390-450 nm Bereich umsetzt. Die Photomultiplier sind so angebracht, das sie direkt am Mineralölgemisch anliegen. Jedes der drei zylinderförmigen Module wird mit zwei Photomultipliern ausgelesen. Die Signale werden über ein Widerstandsnetzwerk aufsummiert.

Insgesamt werden sechs Hochspannungen für die Photomultiplier benötigt, die von einer Hochspannungsversorgung bereitgestellt werden. Zur Zeit wird ein neuer, verbesserter Gamma-Veto-Detektor entwickelt [16].

[8] mit aluminisierter Mylarfolie beschichtet

2.8 Der Beam-Scanner

Direkt vor dem Gamma-Veto-Detektor ist ein System zur Bestimmung der Lage und des Profils des Photonenstrahls aufgebaut. Es handelt sich hierbei um zwei Szintillatoren, von denen einer horizontal und der andere vertikal montiert ist. Die beiden Szintillatoren sind auf einer 45°-Ebene angebracht und werden durch einen Schrittmotor bewegt. Bei einem Scan des Strahlprofils fährt der Beam-Scanner von links unten nach rechts oben durch den γ-Strahl und misst an den einzelnen Positionen für eine festgelegte Zeit die Intensität an dieser Stelle. Da die beiden Szintillatoren nacheinander das Strahlprofil passieren, kann man diese koppeln und benötigt somit zur Auslese nur einen einzigen Photomultiplier. Die Hochspannung für diesen Photomultiplier liefert die Gamma-Veto-HV[9]. Das Signal des Photomultipliers wird auf einen Zähler gegeben.

Bei einer Messung fährt zunächst der vertikal angebrachte Szintillator durch den Strahl und misst damit die horizontale Strahlausdehung. Danach fährt der horizontale Szintillator durch den Strahl, so dass die vertikale Ausdehnung erfasst wird. Dadurch erhält man ein Profil des Photonenstrahls vor dem Gamma-Veto-Detektor.

2.9 Das Triggersystem

Neben der Datenakquisition (Kapitel 2.10) ist der Hardwaretrigger das zentrale Element zur Steuerung der Datennahme. Es wird dabei zwischen zwei Triggerstufen unterschieden. Die Detektorkomponenten, die schnell Informationen liefern, bilden die erste Triggerstufe. Dazu gehören das Taggingsystem, der Gamma-Veto-Detektor in Anti-Koinzidenz, der Innendetektor und das Flugzeitspektrometer. Mit dem Taggingssystem und dem Gamma-Veto-Detektor können zunächst Entscheidungen darüber gefällt werden, ob ein Bremsstrahlungsphoton erzeugt wurde und ob es das System ungehindert passiert hat oder nicht. Wird ein Bremsstrahlungsphoton erzeugt und im Gamma-Veto-Detektor registriert, so kann das Ereignis verworfen werden. Deshalb spricht man von einer Anti-Koinzidenz zwischen Taggingssystem und Gamma-Veto-Detektor. Mit Hilfe des TOF-Flugzeitspektrometers oder des Innendetektors können nun noch weitere Ereignisse selektiert werden.

Die zweite Triggerstufe besteht aus dem **F**ast-**C**luster-**E**n**C**oder (FACE), welcher die Zahl der detektierten Cluster im Crystal-Barrel-Kalorimeter bestimmt. Diese zweite Stufe erlaubt bereits während der Datennahme die Vorselektion spezieller Ereignisklassen.

2.10 Datenakquisition

Die Datenakquisition [17] (kurz: DAQ) speichert die von den einzelnen Detektoren gewonnen Daten. Die Daten der Subdetektoren werden zunächst von lokalen Eventbuildern unter dem Echtzeitbetriebssystem OS/9 gewonnen und dann an einen globalen Eventbuilder übergeben, der die Daten zu Ereignissen zusammensetzt. Die Speicherung der Daten wird vom Eventsaver vorgenommen. Der Datentransfer zwischen globalem Eventbuilder und Eventsaver wird

[9] Hochspannungsversorgung

über eine FDDI Glasfaseranbindung realisiert. Die Daten werden mit dem TCP/IP[10] Protokoll übermittelt und im ZEBRA-Format[11] auf Magnetband oder Festplatte geschrieben.

Die Software des Eventsavers ist in der Programmiersprache C/C++ geschrieben und läuft auf einem Intel-Doppelprozessorsystem unter dem Betriebssystem Linux. Die Steuerung der Datenakquisition erfolgt über die Runcontrol [18], die es dem Benutzer ermöglicht, verschiedene Ausgabemedien und unterschiedliche Trigger zu wählen. Ausserdem ermöglicht sie es, Messungen mit verschiedenen Detektorkomponenten durchzuführen.

2.11 Detektorelektronik

Jeder Sub-Detektor benötigt für die Datenerfassung eine eigene Detektorelektronik. Die gesamte Elektronik des Crystal-Barrel-Kalorimeters ist in der PHOENICS-Halle untergebracht, ebenso wie Teile der zentralen Triggerlogik, die FACE[12], die lokalen Eventbuilder der einzelnen Subdetektoren und der globale Eventbuilder. Die Ausleseelektronik für den Tagger, den Innendetektor und das Lichtpulsersystem befindet sich in der SAPHIR-Area.

Desweiteren stehen an verschiedenen Stellen noch Elektronikracks für die Auslese des Gamma-Veto-Detektors, des Flugzeitspektrometers und für die Kontrolle und Steuerung des Flüssigwasserstofftargets. Ebenso sind Teile des Triggersystems nahe an den einzelnen Detektorkomponenten aufgebaut, um lange Signallaufzeiten zu vermeiden.

[10] Transmission Control Protocol/Internet Protocol

[11] Standard Datenformat für die Hochenergiephysik - entwickelt am CERN für die Programmiersprache Fortran 77

[12] Fast-Cluster-Encoder

Kapitel 3

Anforderungen und Vorgaben

In diesem Kapitel sollen die verschiedenen Anforderungen an eine Überwachung und Steuerung des Crystal-Barrel-Experiments aufgeführt werden. Dies umfasst eine Auflistung der Komponenten, die zu einer möglichst vollständigen Überwachung des Experiments notwendig sind.

3.1 Zu überwachende Komponenten

Im Kapitel 2 wurden die verschiedenen Teile des gesamten Experiments beschrieben. Hier werden nun explizit alle Hardwarekomponenten aufgeführt, für die eine möglichst vollständige Überwachung des Gesamtsystems notwendig und sinnvoll sind.

3.1.1 Hochspannungsversorgungen

Fast jede Detektorkomponente benötigt eine Hochspannungsversorgung für die Szintillatoren und/oder Drahtkammern. Die verschiedenen HVs sind jeweils in der Nähe der einzelnen Detektoren untergebracht. Für die Hochspannungen der Photomultiplier werden ausschließlich Geräte der Firma Le Croy verwendet, die über eine RS-232 oder TTY-Schnittstelle angesprochen werden können. Die Hochspannungsversorgung der Drahtkammer des Taggingsystems wird über ein Gerät der Firma Wiener realisiert.

Im Experiment werden die folgenden Hochspannungsversorgungen verwendet:

Detektor	HV Firma/Typ	Schnittstelle
Gamma-Veto und Beam-Scanner	Le Croy HV 4032A	TTY - max. 300 bps
Tagger-Szintillatoren	Le Croy HV 4032A	TTY - max. 300 bps
Flugzeitspektrometer	Le Croy HV 1445	RS-232 - max. 9600 bps
Innendetektor	Le Croy HV 1450	RS-232 - max. 19200 bps
Taggerkammern	Wiener	Analog I/O - Übers. 1:1000

Tabelle 3.1: Im Experiment verwendete Hochspannungsversorgungen

Die HV-Typen der Firma Le Croy haben unterschiedliche Kommandosätze, so dass jede Hochspannungsversorgung eine eigene Implementierung benötigt.

Für die verschiedenen HVs sollten in der Slowcontrol die folgenden Funktionen implementiert werden, um eine komplette Steuerung und Überwachung zu erreichen:

- Setzen der Spannungen (demand voltages)
- Auslesen der aktuellen Spannungen (current voltages) und der Ströme
- Ein/Ausschalten der Hochspannung
- Fehlermeldung bei Differenz zwischen gesetztem und gemessenem Spannungswert
- Fehlermeldung bei Funktionsstörungen einzelner Kanäle
- Laden und Speichern bestimmter Spannungskonfigurationen
- Anzeige aller gemessenen und gesetzten Spannungen und des jeweiligen Status (Ein/Aus und Fehler)

Für die Hochspannungsversorgung der Taggerkammer wird außerdem noch eine Überwachung des Stromflusses benötigt. Ein zu hoher Strom in der Taggerkammer könnte diese beschädigen. Wird ein überhöhter Strom festgestellt, so muss die Slowcontrol innerhalb kurzer Zeit die Spannung für die Kammern soweit herabsetzen, dass keine Gefahr mehr für sie besteht. Hier muss die Slowcontrol aktiv ohne Benutzerinteraktion in das Experiment eingreifen, um Schäden an der Kammerhardware zu verhindern.

3.1.2 Elektronikcrates

Die vielen verschiedenen Elektronikkomponenten (ADCs, Shaper, Diskriminatoren,...) sind jeweils in 19-Zoll-Racks zusammengefasst. Um auf engem Raum möglichst viel Elektronik unterbringen zu können, haben sich seit vielen Jahren modulare Systeme bewährt, die einen genormten Überrahmen (Crate) verwenden, in den die einzelnen Module gesteckt werden. Diese Crates besitzen eine zentrale Spannungsversorgung und häufig auch ein integriertes Bussystem, mit dem Daten zwischen den einzelnen Komponenten ausgetauscht werden können. Hierfür gibt es unterschiedliche Standards. Das CB-ELSA-Experiment verwendet die vier Standards NIM, CAMAC, FastBus und VME.

Standard	Steckkontakte	Spannungen	Monitorausgang
NIM	7	±6, ±12 und ±24 V	teilw. Centronics
CAMAC	86	±6, ±12 und ±24 V	teilw. Centronics
VME	128	+5, ±12 und ±24 V	-
FastBus	130	5, -5.2, -2 und ±15 V	37-polig Sub-D

Tabelle 3.2: Verwendete Elektronikcrates-Standards

Es gibt verschiedene Möglichkeiten, den Ausfall der Spannungsversorgungen von einzelnen Crates festzustellen. Einige Cratehersteller erlauben es, über einen Monitorausgang den Status der Crates direkt abzufragen, wobei hier bereits alle Spannungen und die Lüfter der Crates automatisch überwacht werden.

Für die Überwachung der Crates wurde schon am CB-LEAR-Experiment ein System mit Spannungskontrolleinheiten und einem Multiplexer entwickelt, welches idealerweise bei CB-ELSA weiterverwendet werden sollte. Der Multiplexer liefert einen digitalen 8-bit-Ausgang mit TTL-Logik[1] und besitzt einen digitalen 3-bit-Eingang. Ingesamt können also $2^3 \cdot 8 = 64$ Crates mit diesem System überwacht werden. Die 64 Eingänge des Controllers benötigen ein optisches Signal, welches über Plastikfaser-Lichtwellenleiter zugeführt werden kann. Für die NIM- und CAMAC-Crates sind entsprechende Module in ausreichender Stückzahl vorhanden.

Für die Fastbus- und VME-Crates stehen keine Module zur Verfügung. Diese müssen im Rahmen dieser Arbeit entwickelt werden. Die Fastbus-Crates verfügen an der Frontseite über eine 37-polige Sub-D Buchse, an der ein Statussignal abgegriffen werden kann. Die VME-Crates besitzen keinen solchen Statusausgang.

3.1.3 Flüssigwasserstoff-Target

Beim Flüssigwasserstoff-Target gibt es eine große Anzahl verschiedener Parameter, die an den diversen Messinstrumenten vor Ort abgelesen werden können. Als analoges Spannungssignal liegt der Pegelstand im Reservoir vor, ebenso wie der Zellendruck der Targetzelle. Alle weiteren Parameter sind nur über Druckanzeigen zugänglich, die nicht ohne größeren Aufwand digitalisierbar sind, so dass sie für eine rechnergestützte Überwachung nicht in Frage kommen.

Da das Target über ein eigenes Interlocksystem verfügt, ist es nicht notwendig, alle Parameter zu überwachen. Um den Status des Targets anzuzeigen, reicht es aus, die Signale der Leveldiode (und damit den Pegelstand im Reservoir) sowie den Zellendruck darzustellen.

3.1.4 FACE

Der **F**ast-**C**luster-**En****C**oder ist Teil der zweiten Triggerstufe. Neben den Diskriminator- und Latchcrates, die durch das Crate-Kontrollsystem überwacht werden, kann noch ein Parameter für die Schwelle zur Clustererkennung gesetzt werden. Diese Schwelle ist der Energiewert in MeV[2], bei dem der FACE-Trigger einen Cluster im Crystal-Barrel-Kalorimeter als Treffer eines Photons erkennen soll. Die Schwellwertenergie wird über ein analoges Spannungssignal, das von einem Netzgerät erzeugt wird, in die FACE-Elektronik eingespeist. Die Energie zur Erkennung eines Clusters ist hierbei proportional zur angelegten Spannung.

Die Spannung sollte idealerweise aus dem Kontrollraum des Experiments eingestellt werden können, um je nach experimentellen Bedingungen unterschiedliche Schwellen zu setzen.

3.1.5 Taggerkammern

Neben den Hochspannungen, die durch eine Hochspannungskontrolle gesetzt und überwacht werden können, benötigen die Drahtkammern des Taggingsystems einen kontinuierlichen Gas-

[1] Transistor-Transitor Logik
[2] Megaelektronenvolt

fluss, um ordnungsgemäß zu funktionieren. Der Gasfluss kann mit Hilfe eines Messgerätes bestimmt werden, das zusätzlich zu einer Anzeige noch einen analogen Ausgang bereithält.

3.1.6 Beam-Scanner

Der Beam-Scanner besteht (wie schon in Kapitel 2.8 beschrieben) aus einem Schrittmotor mit CNC-Ansteuerung via RS-232 Schnittstelle und einem Scaler[3], der ebenfalls über eine solche Schnittstelle ausgelesen und gesteuert werden kann.

Gerät	Firma	Typenbezeichnung	Schnittstelle
Scaler	Heidt Industrie Technik	SC-8000	RS-232
Schrittmotorsteuerung	Isel Automation	IT 116	RS-232

Tabelle 3.3: Geräte des Beam-Scanners

3.2 Anbindung an die Datenakquisition

Um die vom Slowcontrol-System gewonnenen Daten zusammen mit den Experimentdaten auf Magnetband abspeichern zu können, müssen sie an das Datenakquisitionssystem übergeben werden. Zu diesem Zweck müssen die Daten an geeigneter Stelle zentral gesammelt und dort aufbereitet werden. Der Eventsaver (Kapitel 2.10) der Datenakquisition, der die Experimentdaten auf Magnetband oder Festplatte speichert, ist in der Programmiersprache C/C++ geschrieben. Idealerweise sollte die Experimentüberwachung eine C++ Klasse zur Verfügung stellen, die dem DAQ-System die Daten in geeigneter Form liefert.

3.3 Slowcontrol-Datenakquisition, Bussysteme und Masseschleifen

Die verschiedenen von der Slowcontrol zu überwachenden und zu steuernden Geräte haben unterschiedliche Interfaces, die in die folgenden Klassen unterteilt werden können:

- RS-232- und TTY[4]-Schnittstellen
- digitale Ein- und Ausgänge (TTL)
- analoge Ein- und Ausgänge (0 bis 10 Volt)

Die verwendeten Bussysteme müssen diese verschiedenen Interfaces ansprechen können. Zu berücksichtigen ist hierbei vor allem die räumliche Verteilung der Geräte.

Der Aufbau der Detektorkomponenten ist so ausgelegt, dass die einzelnen Detektoren galvanisch voneinander getrennt sind. Es gibt verschiedene Möglichkeiten, dies zu realisieren (siehe [19]). Mit Hilfe dieser Techniken soll vermieden werden, dass kleine Detektor-Signale von Hochfrequenz-Signalen oder 50 Hz Netzspannungen beeinflusst werden. Bei der Installation eines Überwachungs- und Kontrollsystems ist dies unbedingt zu berücksichtigen.

[3] Zähler
[4] current loop

3.4 Benutzerinterface

Der Operator muss durch das Benutzerinterface jederzeit den Status des Detektorsystems erfahren und die Steuerung vom Kontrollraum des Experiments aus vornehmen können. Bei Ausfall von Komponenten soll der Benutzer in geeigneter Weise darüber informiert werden, z.B. durch einen akustischen Alarm.

Außerdem wäre es wünschenswert, den Status des Systems auch an anderen Stellen anzeigen zu lassen. Zur Diagnose sollte die Möglichkeit bestehen, einzelne Komponenten auch von Text-Terminals zu bedienen. Die Steuerung der Hochspannungsversorgungen der Photomultiplier könnte so von Terminals aus durchgeführt werden, die direkt an den Sub-Detektoren aufgebaut sind. Die Einstellung der Hochspannungen kann damit direkt vor Ort erfolgen. Um Bedienungsfehler zu vermeiden, sind bei der Wahl von Spannungen Grenzwerte einzuhalten.

Die Anpassung des Benutzerinterfaces an neue Detektorkomponenten oder veränderte Parameter sollte schnell und ohne großen Aufwand möglich sein, ebenso wie die Programmierung des Slowcontrol-Systems.

Kapitel 4

Die Hard- und Software

In diesem Kapitel soll die für die Experimentüberwachung ausgewählte Hard- und Software vorgestellt und ein Überblick über die gesamte Implementierung gegeben werden.

Da das Überwachungs- und Kontrollsystem Daten von allen Detektorkomponenten benötigt, stellt sich die Frage, wie am einfachsten eine galvanische Trennung erreicht werden kann. Die hier gewählte Möglichkeit ist die Verwendung von rein optischen Übertragungsmedien. Wie dies verwirklicht wurde, soll nun kurz beschrieben werden.

4.1 RS-232- und TTY-Interfaces

Um RS-232- und TTY-Signale über optische Medien übertragen zu können, müssen diese mit Hilfe von Medienkonvertern umgesetzt werden. Hierfür standen Module der Firma AT&T zur Verfügung, die auch für den Anschluss von Terminals und lokalen Eventbuildern verwendet werden. Die Versorgungsspannung von ±12 Volt für die Module wird aus den Elektronikcrates zugeführt. Zu diesem Zweck wurden die Konvertermodule in NIM-Module eingebaut. Die TTY-Signale mussten mit Optokopplern in RS-232-Signale konvertiert werden. Dazu wurde eine kleine Konverterplatine erstellt. TTY (oder auch current loop) bezeichnet einen Übertragungsstandard, bei dem Signale in Form von Strömen übertragen werden, im Gegensatz zu RS-232-Signalen, deren Übertragung mittels Spannungspegeln erfolgt.

Gerät	Position (Subdetektor)	Schnittstelle
Le Croy HV 4032A	Gamma-Veto	TTY
Le Croy HV 4032A	Tagger	TTY
Le Croy HV 1445	TOF	RS-232
Le Croy HV 1450	Innendetektor	RS-232
Scaler HIT SC8000	Gamma-Veto	RS-232
CNC Motorsteuerung Isel IT 116	Gamma-Veto	RS-232
Hall-Sonde MPS FH 54	Taggermagnet	RS-232

Tabelle 4.1: Geräte am Terminal-Server

Um nun nicht für jedes dieser 7 Geräte eine eigene RS-232-Schnittstelle an einem PC verwenden zu müssen, wurden die Geräte an einen Terminal-Server der Firma Lantronix angeschlossen. Der Terminal-Server verfügt über 16 RS-232 Anschlüsse und einen Ethernetnanschluss.

Der Zugriff auf die angeschlossenen Geräte kann nun über den Internet-Standard TCP/IP[1] erfolgen.

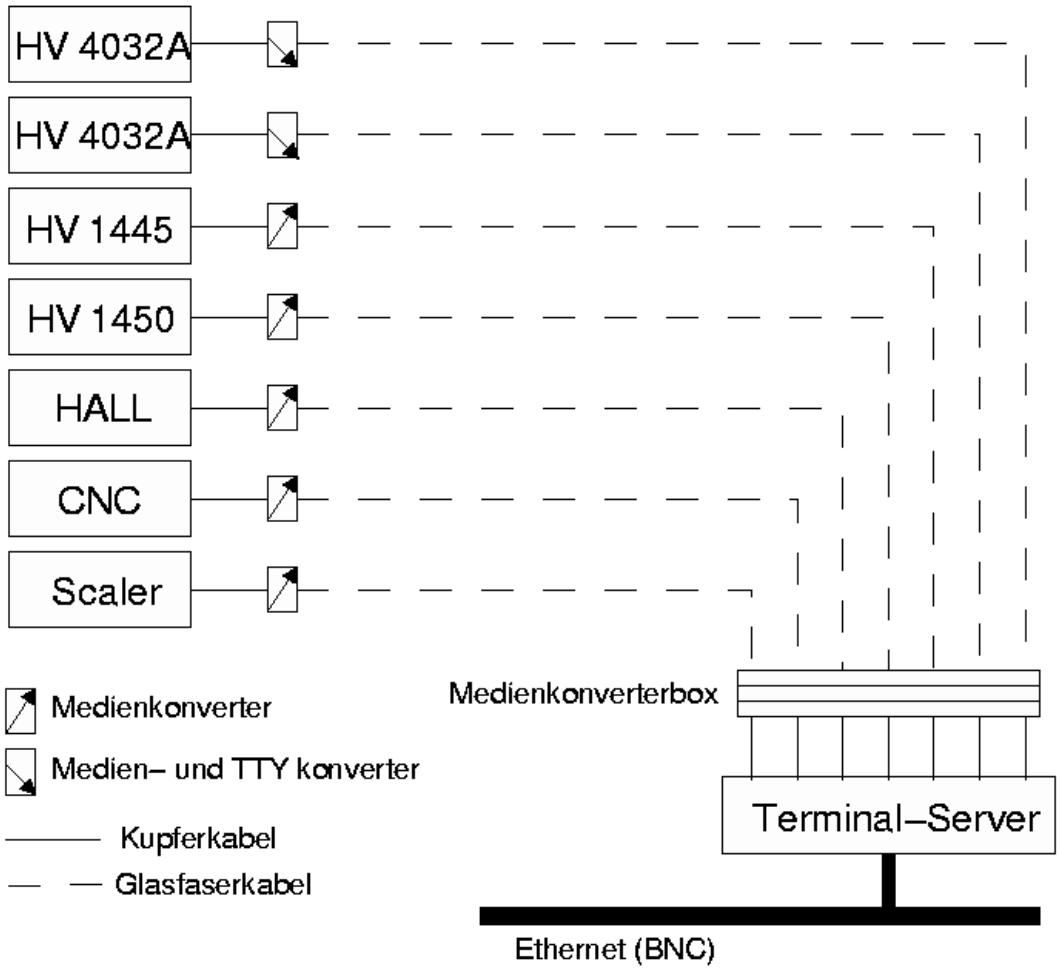

Abbildung 4.1: Terminal-Server und RS-232/TTY-Geräte

Die RS-232-Ports am Terminal-Server werden dabei folgendermaßen auf TCP-Ports abgebildet:

RS-232-Port	TCP-Port
1	2001
2	2002
...	...
16	2016

Tabelle 4.2: Zusammenhang zwischen RS-232-Ports und TCP-Ports am Terminal-Server

Der Zugriff auf den Terminal-Server und damit auf die angeschlossenen Geräte ist somit von jedem Rechner im Experiment mit Ethernetanschluss aus möglich. Das TCP-Protokoll beinhaltet außerdem eine automatische Fehlerkorrektur.

[1]Transmission control protocol / Internet protocol

4.2 Das Feldbussystem „Profibus“

Um auf die verschiedenen analogen und digtalen Ein- bzw. Ausgänge der Geräte zugreifen zu können, wird ein Bussystem benötigt, welches die galvanische Trennung zwischen den Subdetektoren nicht aufhebt und trotzdem einen Datenaustausch mit den verteilten Geräten durchführen kann. Hier soll nun das verwendete Feldbussystem „Profibus“ beschrieben werden, das diese Anforderungen erfüllt.

4.2.1 Feldbussysteme allgemein

Ein Feldbussystem verbindet Mess- und Steuerungsgeräte zu einem System. Feldbussysteme sind serielle Bussysteme, die Daten digital übertragen. In der Industrie werden Feldbussysteme zur Steuerung und Überwachung von Produktionsprozessen eingesetzt. Solche Systeme sind dort weit verbreitet. Insbesondere in der Automobilindustrie, aber auch in der Chemischen Industrie oder in der Lebensmittelproduktion werden Feldbusse eingesetzt.

Vorteile von Feldbussystemen sind:

- verteilte Steuerung und Kontrolle von Geräten und Produktionsanlagen
- Interaktion von verschiedenen Geräteklassen
- geringe Installations- und Betriebskosten
- hohe Zuverlässigkeit

Es gibt verschiedene Feldbusstandards. Die bekanntesten Standards sind:

- Profibus
- CANbus
- Interbus
- DeviceNET

4.2.2 Der Profibus

Für die Slowcontrol wurde ein Profibussystem auf Lichtwellenleiterbasis der Firma Beckhoff Industrie Elektronik ausgewählt. Der Profibus-Standard ist weit verbreitet und findet eine Vielzahl von Anwendungen in der Prozessautomation.

Profibus ist ein lineares Bussystem mit aktivem Busabschluss an beiden Enden und RS-485 als Übertragungsstandard. Als Medium wird ein abgeschirmtes und verdrilltes Kabel verwendet. Stichleitungen sind möglich. Der Profibus kann, je nach eingesetzten Geräten, mit Datenraten von 9.6 kBit/s, 19.2 kBit/s, 93.75 kBit/s, 187.5 kBit/s, 500 kBit/s, 1.5 MBit/s und 12 MBit/s betrieben werden.

Für einen Bus sind laut Profibus-Spezifikation bis zu 32 Stationen in einem Segment erlaubt. Es können Buslängen von bis zu 100 m bei 12 MBit/s und 1,2 km bei 93.75 kBit/s erreicht

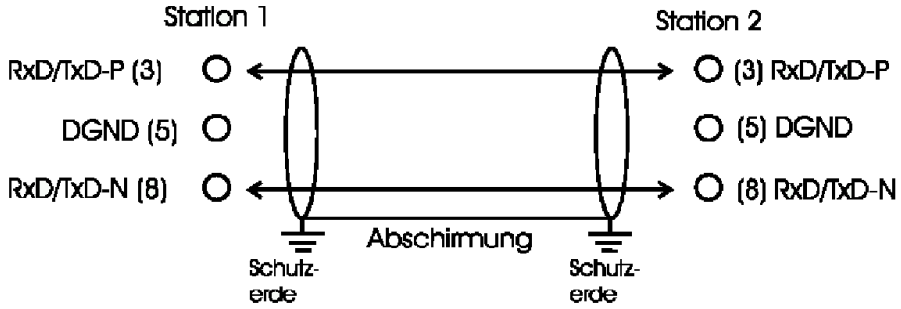

Abbildung 4.2: Profibus-Medium und Terminierung

werden. Mit Repeatern[2] kann die maximale Stationsanzahl auf 127 Stück und die Buslänge auf bis zu 12 km erweitert werden.

Im Profibus-Standard werden zwei Typen von Geräten spezifiziert: Profibus-Master und Profibus-Slaves. Master bestimmen den Datenverkehr auf dem Bus, im Gegensatz zu Slave-Geräten, die nur auf Anforderungen vom Master reagieren und ansonsten passiv bleiben. Master können auch ohne eine externe Aufforderung Daten versenden, wenn sie die Zugriffsberechtigung auf den Bus haben. Sie werden als aktive Komponenten bezeichnet. Neben mehreren Slave-Geräten kann es auch mehrere Profibus-Master in einem Segment geben.

Es gibt verschiedene Protokolle für den Profibus. Der in der Slowcontrol verwendete Betriebsmodus ist der Profibus-DP[3] Modus. FMS[4] ist ein weiterer Betriebsmodus, der allerdings intelligente Feldbusgeräte voraussetzt. Für die Anwendung in der Slowcontrol ist dieser Betriebsmodus nicht interessant, da hier nur die Ansteuerung und Überwachung von verteilten Geräten erforderlich ist.

Der DP-Modus ist für die schnelle Datenübertragung zwischen zentralen Steuergeräten (z.B. speicherprogrammierbaren Steuerungen) und dezentralen Eingangs- und Ausgangsgeräten konzipiert worden. Die Kommunikation erfolgt größtenteils synchron. Die zentrale Steuerung liest Daten vom Bus und schreibt Daten in die Ausgangsregister, um diese an die Ein- und Ausgabegeräte zu senden. Jedes Profibusgerät verfügt über eine Stations-ID. Über diese Identifikationsnummer kann der Profibusmaster jede Station einzeln ansprechen. Außerdem sind im Profibusprotokoll noch Multicasts definiert, die es dem Master erlauben, alle Slave-Geräte gleichzeitig anzusprechen.

Im Profibus-Protokoll sind Mechanismen definiert, um eine Fehlparametrisierung der einzelnen Stationen zu verhindern. Jedem Slave-Gerätetyp ist durch die Profibus-Nutzerorganisation ein eindeutiger Code zugeordnet. Bei der Parametrisierung der Slavegeräte im Profibus-Master muss dieser für jedes angeschlossene Gerät angegeben werden.

Für jeden Slave ist außerdem noch eine Ansprechüberwachung im Master implementiert. Erfolgt innerhalb einer definierten Zeit kein Nutzdatentransport, so wird dies im Profibus-Master festgestellt. Die Slaves verfügen über eine davon unabhängige Überwachung. Wird ein Zeitlimit überschritten, so wird in den gesicherten Betriebsmodus umgeschaltet und der Buskoppler wird als nicht aktiv gekennzeichnet. So werden Ausfälle der Übertragungseinrichtungen erkannt.

[2] Leitungsverstärkern
[3] Decentralized Peripherical
[4] Fieldbus Message Specification

Im Profibus-Standard sind Übertragungsraten von bis zu 12 MBit/s[5] spezifiziert. Das hier verwendete Profibussystem erlaubt Übertragungsraten von 1.5 MBit/s. Als Medium werden Lichtwellenleiter aus Plastikfasern verwendet. Die Anbindung an das Profibussystem geschieht mit einem Optical-Link-Modul der Firma Siemens. Mit diesem Modul kann ein Ringsystem auf Lichtwellenleiterbasis an den lineare Profibus angeschlossen werden:

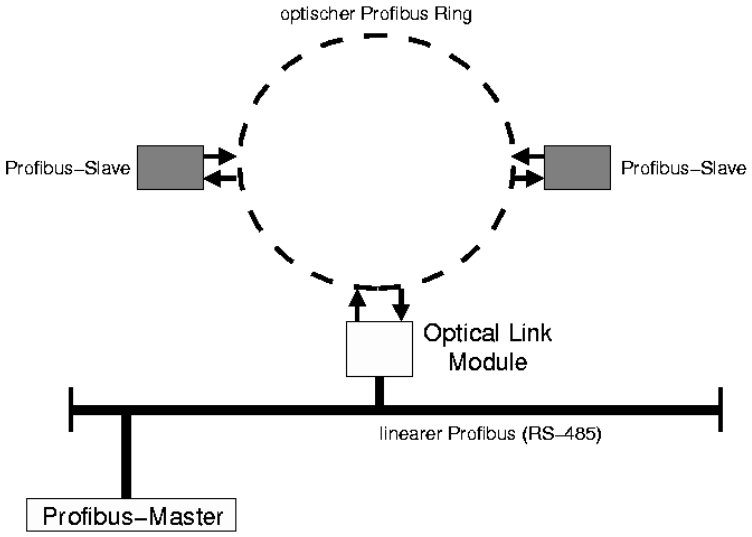

Abbildung 4.3: Optische Profibus-Ringstruktur

Erlaubt sind bei einer Datenübertragungsrate von 1.5 MBit/s maximal 10 Stationen im Ring. Hierbei sind folgende Mindest- und Maximallängen bei den optischen Übertragungsstrecken zu berücksichtigen:

Verbindungstyp	Mindestlänge	Maximallänge
Master → Slave	1 m	34 m
Slave → Slave	1 m	25 m
Slave → Master	0 m	46 m

Tabelle 4.3: Mindest- und Maximallängen in optischen Profibus-Ringsystemen

Für den Einsatz in der Überwachung des Crystal-Barrel-Experiments sind die spezifizierten Maximallängen vollkommen ausreichend. Die maximale Kabellänge, die zwischen der SAPHIR-Area (Experiment) und der PHOENICS-Halle (Elektronik) erreicht wird, beträgt etwa 25 m.

4.2.3 Das System der Firma Beckhoff

Die Profibus-Komponenten der Firma Beckhoff wurden ausgewählt, da das System einen sehr einfachen Datenaustausch zwischen dem System und der noch zu schreibenden Software ermöglicht. Außerdem ist der Aufbau von einzelnen Profibus-Slave-Stationen so ausgelegt, das weitere Module leicht an die Stationen angefügt werden können.

Insgesamt besteht das System aus den folgenden Komponenten:

[5] MBit/s - Millionen bit pro sekunde

- **Profibus Master:** PCI-Controllercard für den Einsatz in Standard-PC Systemen mit RS-485 Anschluss
- **Profibus Slaves:** 8 Buskoppler BK 3500 mit HP-Simplexstecksystem für LWL[6]-Kabel
- diverse Module (Analog/Digital-Wandler, Netzteilklemmen, Busendklemmen)
- Systemsoftware für Microsoft Windows NT, Klemmenkonfigurationssoftware und Softwaremodule zur Anbindung an andere Programme (ADS DLL und ActiveX Objekte)

Jeder der Buskoppler ermöglicht den Anschluss von bis zu 64 Modulen (Busklemmen). Erhältlich sind Analog- und Digital-I/O[7]-Klemmen und solche mit Spezialfunktionen (Winkel- und Wegmessung, Temperaturbestimmung, etc.). Die Buskoppler verfügen über ein internes Kommunikationssystem (K-Bus) zum Austausch von Daten mit den Busklemmen. Terminiert wird der K-Bus über ein Busendterminal.

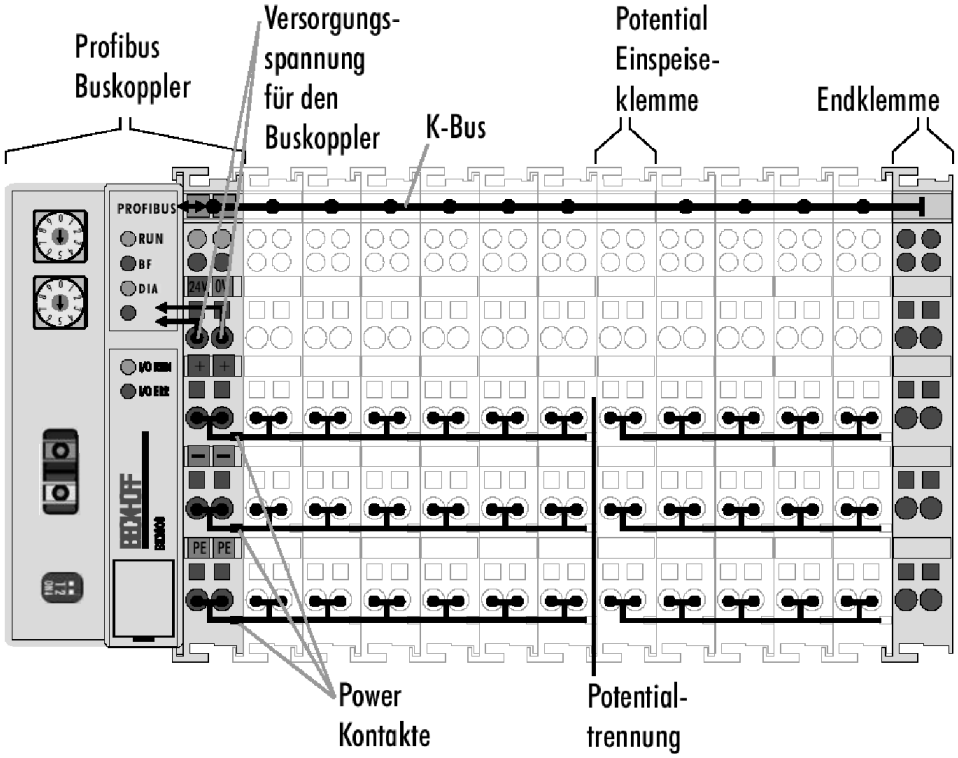

Abbildung 4.4: Buskoppler und I/O-Klemmen des Beckhoff-Systems

Mit zwei Drehschaltern wird am Buskoppler die Stations-ID festgelegt. Jeder Buskoppler verfügt außerdem über Diagnoselampen und einen Feldbusanschluss. Eine Versorgungsspannung von 24 Volt muss dem Koppler mit einem separatem Netzteil zugeführt werden. Das ganze System kann auf einer DIN-Hutschiene montiert werden.

Die Systemsoftware TwinCat-I/O ermöglicht die Parametrisierung des gesamten Profibussystems. Mit dem TwinCat-I/O-Systemmanager wird eine Verknüpfung von Variablen mit

[6] Lichtwellenleiter
[7] Input/Output - Eingabe/Ausgabe

Profibus-Datenregistern durchgeführt. Diese Variablen können über verschiedene Softwareschnittstellen ausgelesen und/oder beschrieben werden, so dass man Parameter der angeschlossenen Geräte lesen und setzen kann. Von den Softwareschnittstellen wird die ADS Dynamic-Link-Library[8] verwendet werden.

4.3 Datenerfassung

Die Slowcontrol besteht aus zwei Programmkomponenten: zum einen aus der mit LabView entwickelten Benutzeroberfläche, zum anderen aus dem Slowcontrol-Dämon als zentraler Instanz für die Datenerfassung und Speicherung. Der Slowcontrol-Dämon ist in der Sprache C/C++ geschrieben und wickelt die Kommunikation mit den RS-232/TTY-Geräten ab. Desweiteren übernimmt er die Daten des Profibussystems vom Benutzerinterface.

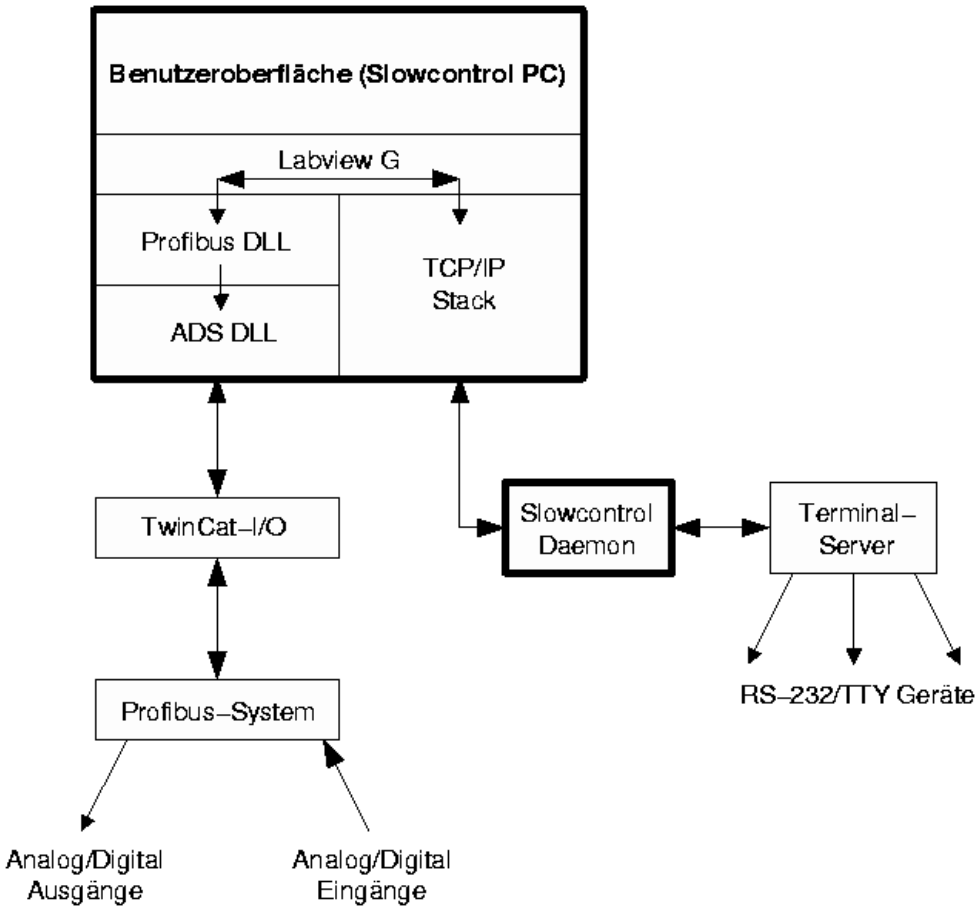

Abbildung 4.5: Datenerfassung der Slowcontrol

Die im Slowcontrol-Dämon gespeicherten Daten können zum Beispiel von der Datenakquisition abgerufen werden, um sie dann zusammen mit den Experimentdaten auf ein Bandlaufwerk oder Festplatte zu schreiben.

Dieses Konzept ermöglicht es, Geräte-Kommandos auch ohne graphische Benutzeroberfläche

[8] Microsoft Windows DLL

abzusetzen und den Status an anderen Orten anzeigen zu lassen. Eine Möglichkeit hierfür ist die bereits erwähnte Steuerung von Hochspannungsversorgungen.

4.4 Benutzerinterface

Das Benutzerinterface des Überwachungs- und Kontrollsystems wird mit einer Visualisierungssoftware der Firma National Instruments und dem Betriebssystem Microsoft Windows NT 4.0 erstellt. Die Software LabView ermöglicht die einfache und schnelle Erstellung von Anzeigen und Kontrollelementen mit einem graphischen Entwicklungssystem, der Programmiersprache G. Die Visualisierungssoftware bietet außerdem diverse Schnittstellen, so dass auf Datenerfassungsgeräte und Datenquellen zugegriffen werden kann. LabView unterstützt die Kommunikation über die IP-Protokolle TCP[9] und UDP[10].

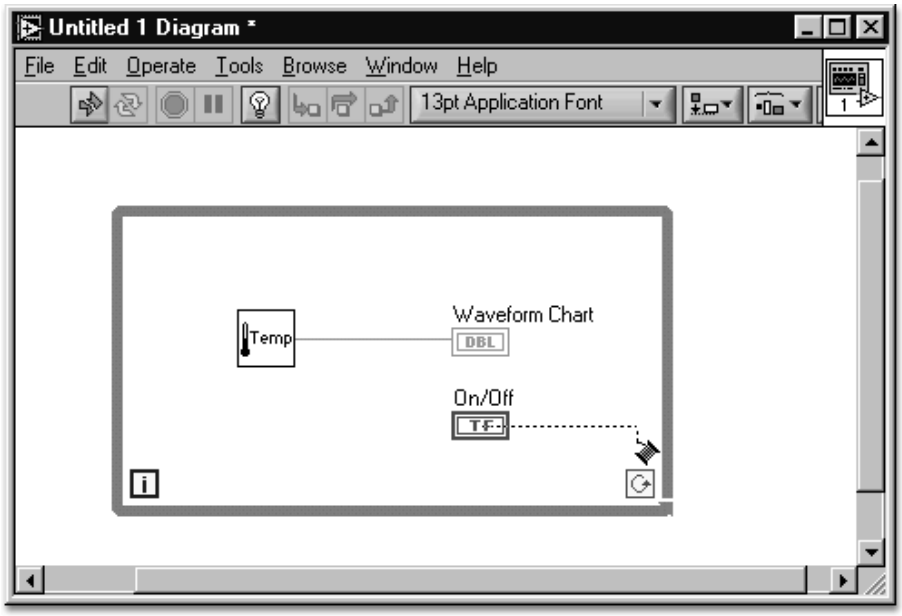

Abbildung 4.6: Beispiel zur Programmierung in LabView mit der Sprache G

4.4.1 Kommunikation mit dem Profibus-System

Zur Kommunikation mit dem Profibussystem bietet LabView die Möglichkeit, Funktions-Aufrufe in das LabView-Programm aufzunehmen. Unter Angabe der Funktion, der Parameter und Rückgabeparameter werden die Funktionen innerhalb einer Funktionsbibliothek aufgerufen. Der Zugriff auf die Profibusdaten wird über einen Bibliotheks-Aufruf via ADS-DLL der Firma Beckhoff ermöglicht. Bei der hier verwendeten Implementation wurde eine weitere DLL als Interface zur LabView-Oberfläche entworfen, um damit einfacher auf die Profibusdaten zugreifen zu können.

[9] Transmission control protocol
[10] User datagram protocol

4.4.2 Kommunikation mit den RS-232/TTY-Geräten

Die Kommunikation mit den RS-232/TTY-Geräten erfolgt - nach dem Anschluss dieser an den Terminal-Server - über TCP/IP. Dies wird vom Slowcontrol-Dämon realisiert, der die Daten der Geräte zwischenspeichert und der diese der Benutzeroberfläche über eine einheitliche Schnittstelle zur Verfügung stellt.

4.5 Entfernte Steuerung und Datenübergabe

Die Kontrolle und Steuerung des Experiments kann lokal vom Slowcontrol-PC vorgenommen werden, der in der PHOENICS-Halle steht. Der Operator kann aber auch vom Crystal-Barrel-Kontrollraum aus auf diesen Rechner zugreifen. Hierzu wird die Software VNC[11] verwendet. Diese erlaubt die Steuerung von Computer-Arbeitsoberflächen in heterogenen Rechnernetzen. Hier ermöglicht sie die Steuerung des Slowcontrol-Rechners[12] von einem Linuxrechner aus. Die Software VNC lässt dabei jedoch nur eine einzige Verbindung für die Steuerung zu.

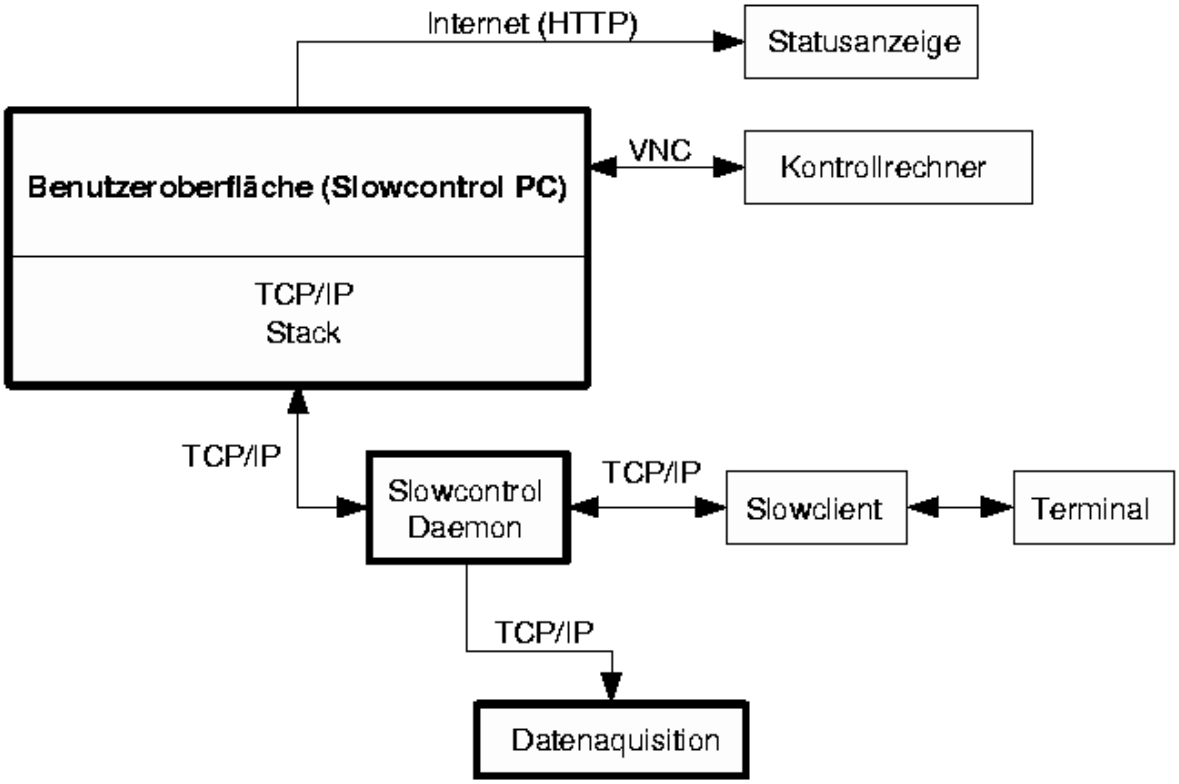

Abbildung 4.7: Zugriff auf die Daten der Slowcontrol

Mit Hilfe eines Webbrowsers kann ein „Schnappschuss“ des Benutzerinterfaces angefertigt und angezeigt werden. Auf diese Weise kann der Status von beliebigen Orten abgerufen werden.

[11] Virtual Network Computing
[12] mit Microsoft Windows NT

Kapitel 5

Der Slowcontrol-Dämon

In diesem Kapitel soll die Umsetzung der zentralen Datenspeicherung und der Kontrolle der Terminal-Server-Geräte beschrieben werden, die im Slowcontrol-Dämon zusammengefasst wurde. Als Dämon bezeichnet man im allgemeinen einen im Hintergrund laufenden Prozess, der Dienste zur Verfügung stellt. Der Slowcontrol-Dämon wurde in der Programmiersprache C/C++ implementiert.

5.1 C++ Klassendiagramm für die Terminal-Server Geräte

Die **O**bjekt**o**rientierte **P**rogrammierung (**OOP**) in der Programmiersprache C++ ermöglicht den modularen Aufbau des Slowcontrol-Dämons. Das Konzept der OOP unter Verwendung von Klassen beruht darauf, dass eine Sammlung von Funktionen und Variablen (eine sogenannte Klasse) vererbt werden kann. Die abgeleiteten Klassen verfügen über die Eigenschaften der ursprünglichen Klasse. Zusätzlich können diese Eigenschaften erweitert, bzw. neue hinzugefügt werden [20].

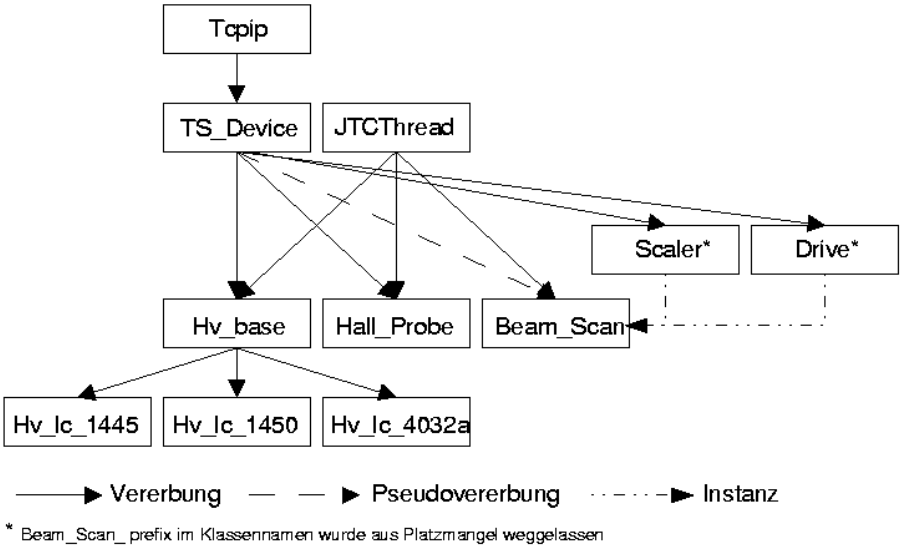

Abbildung 5.1: Klassenstruktur der Slowcontrol-Klassen

Die hier aufgeführten Klassen und ihre Funktion werden im nachfolgenden beschrieben. Neben den im Diagramm abgebildeten Klassen existieren noch weitere, die für die Kommunikation zwischen dem Slowcontrol-Dämon und der DAQ, bzw. dem Benutzerinterface zuständig sind.

5.2 Die Threading-Klasse JTCThread

Um die parallele Verarbeitung von Prozessen in einem Programm zu ermöglichen, ist es notwendig, Multithread-Mechanismen zu implementieren. Multithreading steht für die quasi gleichzeitige Ausführung getrennter Programmteile. Unter Unix-Betriebssystemen gibt es dafür Threads nach dem Posix Standard (pthreads).

Für den Slowcontrol-Dämon wird das Paket JThreads/C++[1] der Firma OOC[2] benutzt. Dies ermöglicht die einfache Verwendung von Threading-Mechanismen in Anlehnung an die Programmiersprache Java. Wie einfach die sonst komplizierte Thread-Programmierung dadurch wird, soll hier kurz an einem Beispiel dargestellt werden:

Beispiel für eine Thread-Klasse

```
#include <JTC/JTC.h>
#include <iostream>
class CMeinThread : public JTCThread {
  public:
    CMeinThread();
    void run();
    void start();
};

CMeinThread::CMeinThread() { }

void CMeinThread::start() {
  JTCThread::start();
}

void CMeinThread::run() {
  while (true) { cout << "Diese Zeile wird in einem Thread ausgegeben" << endl; }
}

void main() {
  JTCInitialize initialize;
  CMeinThread *MeinThread;
  MeinThread = new CMeinThread();
  MeinThread->start();
}
```

Die hier erstellte Klasse **CMeinThread** verfügt über die beiden als public deklarierten Funktionen *start()* und *run()*. Über die Funktion *start()* wird der Thread gestartet, der in der Funktion *run()* implementiert ist. Der ganze Thread wird beendet, sobald die Funktion *run()*

[1] Java-like Threads for C++
[2] Object Oriented Concepts Inc. - http://www.ooc.com/

beendet wurde. Im Beispiel ist eine Endlosschleife implementiert. Dieses Beispiel zeigt, wie leicht sich Thread-Mechanismen mit Hilfe der Objektorientierten Programmierung und der JTCThread-Klasse verwirklichen lassen.

Im Dämon wird für jedes Gerät ein einzelner Thread gestartet, so dass die Datenerfassung für alle Geräte parallel erfolgt. Das Klassendiagramm (Abbildung 5.1) zeigt, welche der Klassen Threading verwenden. Dies sind:

- die Basisklasse für alle Hochspannungsversorgungen **Hv_Base**
- die Klasse zur Abfrage der Hall-Sonde **Hall_Probe**
- die Klasse zur Ansteuerung des Beam-Scanners **Beam_Scan**

Damit erhält jedes Gerät, das durch den Slowcontrol-Dämon abgefragt werden soll, Multi-Thread Fähigkeiten. So wird die parallele Datenerfassung ermöglicht.

5.3 Die TCP/IP-Klasse Tcpip

Die TCP/IP-Kommunikation mit dem Terminal-Server wird durch die Klasse **Tcpip** implementiert. Sie ist Grundlage für alle weiteren Klassen, die Daten vom Terminal-Server verarbeiten. **Tcpip** stellt grundlegende Funktionen für die Kommunikation zur Verfügung:

Die öffentlichen Funktionen der Tcpip Klasse

```
Tcpip(int port, char *address);
Tcpip(int port);
~Tcpip();
void connect();
void disconnect();
void send_data(char *data);
void send_data(char data);
void receive_data(char *buf);
void receive_data(char *buf,int timeout);
void receive_char(char *buf);
void receive_char(char *buf,int timeout);
```

Als Konstruktor und Destruktor werden Funktionen bezeichnet, die beim Erzeugen bzw. Entfernen einer Klasse aufgerufen werden. Der Konstruktor ist für die Initialisierung aller Variablen zuständig. Er trägt den gleichen Namen wie die Klasse selbst. Es sind mehrere Konstruktoren mit verschiedenen Parametern möglich. Der Konstruktor der Klasse **Tcpip** legt die Zieladresse und den Port für die Kommunikation fest.

Die Funktionen *connect()* bzw. *disconnect()* stellen eine Verbindung mit dem Kommunikationspartner via TCP her bzw. beenden diese. Mit Hilfe der Funktionen *send_data()* und *receive_data()* können Daten gesendet oder empfangen werden. Dabei ist es möglich, neben Zeichenketten auch einzelne Zeichen zu senden oder zu empfangen.

Der Destruktor *Tcpip* wird beim Beenden der Klasse aufgerufen und schließt Verbindungen, die noch geöffnet sind.

5.4 RS-232-Geräte und Klasse TS_Device

Zweck der von **Tcpip** abgeleiteten Klasse **TS_Device** ist die Definition eines gemeinsamen Interfaces für alle an den Terminal-Server angeschlossenen Geräte. Dadurch kann die grundlegende Kommunikation mit allen Geräten vereinheitlicht werden. Natürlich erfordern die unterschiedlichen Gerätetypen eigene Implementationen, die dem Befehlssatz und den gelieferten Daten der Geräte gerecht werden. Dies erfolgt in speziellen Klassen, die später beschrieben werden.

Jedem Gerät wird zunächst ein Name zugewiesen, über den dieses Gerät später angesprochen werden kann (Kapitel 5.5). Dafür sind die folgenden Funktionen und Variablen zuständig:

Namenszuweisungen in der Klasse TS_Device

```
string getname();
void setname (string newname);
```

Desweiteren werden als virtuelle Funktionen definiert:

Virtuelle und öffentliche Funktionen der Klasse TS_Device

```
virtual vector <int> getdata()=0;
virtual void push_command(char *command);
virtual void do_cmd();
virtual void stop_cmd();
virtual bool queue_status();
```

Virtuelle Funktionen haben die besondere Eigenschaft, dass sie zwar definiert werden, jedoch durch eine abgeleitete Klasse erst implementiert werden müssen. Die Funktion *getdata()* definiert für alle Geräte ein gemeinsames Interface, mit dessen Hilfe die erfassten Werte abgefragt werden können. Für die einfachere Handhabung wurde auf die Funktionen der Standard-Template-Library zurückgegriffen, welche das Template vector <> definiert. Dadurch wird es möglich, eine unbekannte Anzahl an Werten von der Funktion *getdata()* zu empfangen. Mit Vektoroperationen kann man nun auf diese zugreifen.

Mit der Funktion *push_command()* können Kommandos an einen Kommandopuffer angefügt werden, der zu einem späteren Zeitpunkt an das Gerät übermittelt wird. Die Funktion *queue_status()* fragt den Zustand dieses Kommandobuffers ab[3]. Im Gegensatz dazu wird in der Funktion *do_cmd()* die Ausführung eines einzelnen Befehls implementiert. Die Funktion soll eine Mehrfachausführung von Kommandos verhindern und wird später von der Klasse **Beam_Scan** benutzt (siehe Kapitel 5.4.3). *stop_cmd()* unterbricht die Ausführung des in *do_cmd()* implementierten Befehls.

5.4.1 Hall-Sonde und Klasse Hall_Probe

Die Hall-Sonde im Taggingmagneten ist hinsichtlich ihrer Programmierung eines der am leichtesten einzubindenden Geräte. Die Kommandos zur Abfrage der Temperatur und der Magnetfeldstärke können Tabelle 5.1 entnommen werden.

[3] leer: false wird zurückgeliefert - gefüllt: true wird zurückgeliefert

Kommando	Antwort
?TEMP	Temperatur in °C
?MEAS	Magnetfeld in Tesla oder mTesla

Tabelle 5.1: Kommandosequenzen der Hall-Sonde

Die Funktionen *read_field()* und *read_temperature()* liefern diese Werte als Datentyp float zurück. In der Funktion *run()* werden diese beiden Messwerte automatisch alle zwei Minuten erfasst. Die Funktionen *read_field()* und *read_temperature()* nutzen für die Datenerfassung Funktionen der Klasse **Tcpip**.

Die Klasse ist abgeleitet von der Basisklasse für die Terminal-Servergeräte (**TS_Device**). Die Implementation der Funktion *getdata()* aus der **TS_Device** Klasse liefert als ersten Wert die Magnetfeldstärke in 10^{-3} Tesla und als zweiten Wert die Temperatur in 10^{-1} $°C$ zurück.

5.4.2 HV-Steuerung

Die Ansteuerung der Hochspannungsversorgungen gestaltet sich schwieriger. Es müssen Spannungen ausgelesen und neue Spannungen eingestellt werden können. Zudem muss es dem Benutzer ermöglicht werden, die HV ein- und auszuschalten. Die Hochspannungsversorgungen verfügen über unterschiedliche Kommandosätze, so dass für jeden HV-Typ eine Implementation notwendig ist. Um die Verwendung der HV-Klassen im Dämon zu vereinfachen, wurde in der Klasse **Hv_Base**, die als Basisklasse für jeden Gerätetyp dient, ein einfaches Interface definiert, welches für alle Hochspannungsversorgungen gleich ist. Dieses Interface besteht aus den Funktionen:

Virtuelle und öffentliche Funktionen der Klasse Hv_Base

```
void push_command(char *command);
void run();

virtual vector <int>status()=0;
virtual vector <int>read_volt()=0;
virtual vector <int>demand_volt()=0;

virtual void on()=0;
virtual void off()=0;
virtual void set_volt(int channel,int volt)=0;
```

Die Funktion *push_command()* analysiert eine übergebene Zeichenkette und extrahiert daraus (falls möglich) ein Kommando. Mögliche Kommandos sind **HVON**, **HVOFF** und **SETVOLT**. Dem Kommando **SETVOLT** muss jeweils noch ein Parameter für den gewünschten Kanal und die neu zu setzende Spannung angefügt werden.

Die Befehle werden in einem Kommandopuffer gespeichert und zu gegebener Zeit von der Funktion *run()* ausgeführt. In der Funktion *run()* werden nacheinander die Funktionen *read_volt()*, *demand_volt()* und *status()* aufgerufen, die jeweils die aktuellen Spannungen (current voltages), die geforderten Spannungen (demand voltages) und den Status der Hochspannungsversorgung sowie ihrer Kanäle zurückliefern:

Auszug aus der Funktion run() der Klasse Hv_Base

```
while(running==1) {
  current_voltages=read_volt();
  if (running==0) break;
  check_command_queue();

  demand_voltages=demand_volt();
  if (running==0) break;
  check_command_queue();

  hv_status=status();
  if (running==0) break;
  check_command_queue();
}
```

Nach jedem dieser Aufrufe wird mit Hilfe der Funktion *check_command_queue()* überprüft, ob ein Kommando vom Benutzer in den Kommandopuffer geschrieben wurde. Ist dies der Fall, so werden die Befehle der Reihe nach bearbeitet. Die Funktionen *on()*, *off()* und *set_volt()*, die in den abgeleiteten Klassen implementiert werden müssen, senden die Kommandos an die Hochspannungsversorgungen. Danach werden alle Spannungen von den Hochspannungsversorgungen abgerufen, um einen konsistenten Status zu erhalten. Nach Abarbeitung aller Befehle ist der Kommandopuffer leer.

Die abgeleiteten Klassen müssen nun die als virtuell definierten Funktionen implementieren. Die Kommandosequenzen für die Hochspannungstypen sind in Anhang A aufgeführt.

Die Funktion *getdata()* liefert die aktuellen und gesetzten Spannungen, sowie den Status der HV bzw. zum Teil auch den Status einzelner Kanäle zurück. Die von der Funktion *getdata()* gelieferten Daten haben dabei das folgende Format:

Current voltage Kanal 1
...
Current voltage Kanal n
-99
Demand voltage Kanal 1
...
Demand voltage Kanal n
-99
Status HV an/aus (1 oder 0)
Status Kanal 1 (0 kein Fehler)
...
Status Kanal n (0 kein Fehler)
-99

Tabelle 5.2: Rückgabewerte der Funktion *getdata()* für die Hochspannungsversorgungen

Nicht alle Hochspannungsversorgungen liefern Statusinformationen für die einzelnen Kanäle zurück, sondern nur eine Information über den Zustand der Hochspannung (ein/aus). Fehler

einzelner Kanäle können durch Vergleich der gemessenen Spannung (current voltage) und der gesetzten Spannung (demand voltage) ermittelt werden. Dies geschieht aber erst innerhalb des Benutzerinterfaces.

5.4.3 Beam-Scanner

Die Programmierung des Beam-Scanners unterscheidet sich von der Programmierung der Hall-Sonde oder der Hochspannungsversorgungen, da im Beam-Scanner zwei verschiedene Geräte anzusteuern sind. Die Verwendung einer einzigen Klasse ist hier nicht möglich.

Um nun die Vorteile der Objektorientierten Programmierung zu nutzen, wurde hier folgendermaßen vorgegangen: Die Klassen **Beam_Scan_Scaler** und **Beam_Scan_Drive** sind von **TS_Device** abgeleitete Klassen. Die Klassen erhalten dabei Eigenschaften und Funktionen, mit denen sie auf Terminal-Server-Geräte zugreifen können. Im Unterschied zu anderen Klassen sind sie nicht von der Klasse **JTCThread** abgeleitet. Dies wäre nicht sinnvoll, da die Klassen dann als unabhängige Prozesse liefen. Schrittmotor und Zähler müssen aber aufeinander abgestimmt sein, da der Schrittmotor die Szintillatoren an eine bestimmte Position fahren muss, um dort für eine definierte Zeit eine Zählrate zu ermitteln.

Um die Daten zu erfassen und an den Slowcontrol-Dämon weiterzugeben, wurde die Klasse **Beam_Scan** von der Klasse **TS_Device** abgeleitet. Somit erhält **Beam_Scan** die Eigenschaften für die Datenübergabe an den Dämon (Funktion *getdata()*). Durch die Klasse wird keinerlei Zugriff auf die Geräte am Terminal-Server direkt vorgenommen. Dies geschieht mit Hilfe von Instanzen der Klassen **Beam_Scan_Scaler** und **Beam_Scan_Drive**. Zu sehen ist dies im Klassendiagramm (Pseudovererbung), da nur ein kleiner Teil der Funktionalität der Klasse Verwendung fand.

Die Klasse **Beam_Scan_Scaler** stellt die Funktionen für den Zugriff auf den Scaler des Beam-Scanners zur Verfügung:

Öffentliche Funktionen der Klasse Beam_Scan_Scaler

```
void reset();
double read();
vector <int>getdata();
void init_scaler();
```

Die Funktion *reset()* löscht den Zählerstand des Scalers; *read()* liest den Zähler aus und liefert die zum Tagger-OR[4] normierte Zählrate zurück. Die Implementation der Funktion *getdata()* liefert nur einen Wert (Null) zurück, da die Datenübergabe an den Dämon von der Klasse **Beam_Scan** durchgeführt wird. *init_scaler()* initialisiert den Zähler und überprüft dessen Konfiguration.

Die Ansteuerung des Schrittmotors wird über die Klasse **Beam_Scan_Drive** realisiert. Sie enthält die Funktionen:

[4] Logisches OR der 14 Tagger-Szintillatoren

Öffentliche Funktionen der Klasse Beam_Scan_Drive

```
int getPosition();
void setPosition(int newpos);
void init_drive();
vector <int> getdata();
```

setPosition() veranlasst den Schrittmotor, an eine neue Position zu fahren, die der Funktion als Parameter übergeben wird. Die momentane Position kann mit der Funktion *getPosition()* ermittelt werden. *getdata()* implementiert ebenfalls keine Datenübergabe (siehe **Beam_Scan_Scaler**).

Die Klasse **Beam_Scan** führt die Vermessung des Strahlprofils durch:

Öffentliche Funktionen der Klasse Beam_Scan

```
void run();
void do_cmd();
void stop_cmd();
vector <int> getdata();
```

In der Funktion *run()* wird auf den externen Aufruf der Funktion *do_cmd()* gewartet. Dies sollte durch den Slowcontrol-Dämon erfolgen, wenn der Benutzer einen Beam-Scan durchführen möchte. Durch Implementation der Funktion *do_cmd()* wird verhindert, dass ein Benutzer mehrere Beam-Scans nacheinander anfordert. Aus diesem Grund wurde hier kein Kommandopuffer implementiert, der Befehle speichert und nacheinander ausführt. Mit der Funktion *stop_cmd* kann ein gestarteter Beam-Scan abgebrochen werden. Die Funktion *getdata()* liefert die Daten des Photonenstrahlprofils in der folgenden Form zurück:

Position 1
Rate 1
Position 2
Rate 2
..
..
Position n
Rate n
-98/-99

Tabelle 5.3: Von *getdata()* zurückgelieferte Daten für den Beam-Scanner

Insgesamt enthält der zurückgegebene Vektor für jeden der n Datenpunkte Position und Zählrate. Abgeschlossen wird der Vektor mit einem negativen Wert, der später im TCP-Server (siehe Kapitel 5.5) als Endmarkierung dient. Wird zum Zeitpunkt des Aufrufs der Funktion *getdata()* ein Beam-Scan durchgeführt, so erhält man den Zahlenwert -98, sonst ist der letzte Wert -99. Dadurch kann das Benutzerinterface diese Information darstellen. Alle zwei Stunden wird ohne Aufforderung durch einen Benutzer eine Vermessung des Strahlprofils durchgeführt, um die Position des Strahls festzuhalten und zu überwachen.

5.5 TCP-Server-Klasse

Zum Datenaustausch mit externen Programmen (DAQ, Profibussystem oder Benutzerinterface) muss ein einheitliches Interface definiert werden, mit dessen Hilfe auf die gespeicherten Daten zugegriffen werden kann. Für den Slowcontrol-Serverdienst wurde zu diesem Zweck ein einfaches, auf TCP/IP basierendes Protokoll festgelegt. Nach einem erfolgten Verbindungsaufbau wartet der Server auf einen Befehl oder eine Anfrage des Benutzerprogramms (Client). Die Anfrage des Clients besteht dabei aus dem folgenden Befehlssatz:

GET	<Gerätename>	fordert Daten vom Gerät an
SET	<Gerätename> <Kommando>	schickt Kommando an Gerät
DO	<Gerätename>	führt einen speziellen Befehl aus
STOP	<Gerätename>	bricht Ausführung eines spez. Befehls ab
STORE	<Datenname>	speichert Daten unter Namen ab
QUEUESTATUS	<Gerätename>	erfragt Status des Kommandopuffers

Tabelle 5.4: Kommandos des Slowcontrol-Servers

Der Server antwortet auf jeden dieser Befehle mit einer einzigen Zeile. Sind die Kommandos **SET**, **DO** und **STOP** in der richtigen Syntax, so wird dies mit einem **OK** beantwortet. Auf das Kommando **STORE** anwortet der Slowcontrol-Dämon mit **STORED**, wenn die Information erfolgreich abgespeichert werden konnte.

DO bewirkt, dass die Funktion *do_cmd()* in der Klasse des jeweiligen Geräts aufgerufen wird. Das Kommando **STOP** bricht die Ausführung dieses Kommando ab, indem die Funktion *stop_cmd()* aufgerufen wird. Der Befehl **GET** liefert die zum entsprechenden Gerät gehörenden Daten. Dabei bedient sich der TCP-Server der Funktion *getdata()*, falls es sich hierbei um ein Gerät des Terminal-Servers handelt. Die einzelnen Integerwerte werden dabei hintereinander in einer Zeile ausgegeben. Hier eine typische Anforderung mit Hilfe des **GET** Befehls:

Beispiel für die Abfrage der Tagger HV

```
GET TAGGER
1425 1370 1400 1470 16 1450 1544 1509 1435 1415 1410 1450 1450 1420 1330
1290 1330 1290 1312 1326 1406 1390 1436 1407 1430 1480 1350 1241 1483 247
7 8 -99 1425 1370 1400 1470 0 1450 1544 1509 1435 1415 1410 1450 1450 1420
1330 1290 1330 1290 1312 1326 1406 1390 1436 1407 1430 1480 1350 1240 1483
0 0 0 0 -99 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -99
```

Liegen noch keinerlei Informationen für das angeschlossene Gerät vor (z.B. direkt nach dem Start des Slowcontrol-Dämon), so wird -1 zurückgeliefert.

Eine andere Möglichkeit für das **GET** Kommando ist der Abruf von Daten, die über das Kommando **STORE** dort gespeichert wurden. Das Kommando **GET** liefert dann die Zeichenkette zurück, die dem Kommando **STORE** zuvor übergeben wurde. Hier ein Beispiel für nacheinander ausgeführte **STORE** und **GET** Befehle:

Speicherung von beliebigen Daten im Slowcontrol-Dämon

```
STORE MEINEDATEN 1 2 3 4 5
STORED
GET MEINEDATEN
1 2 3 4 5
```

Das Benutzerinterface verwendet den **STORE** Befehl, um Daten von Profibus-Geräten abzulegen, um diese der Datenakquisition zur Verfügung zu stellen. Neben dieser Anwendung des **STORE** Kommandos ist es aber auch möglich, Daten von anderen externen Datenquellen im Dämon zu speichern. Dies wird später relevant, wenn neue Detektoren mit eigenem Slowcontrol-System in das bestehende Experiment integriert werden (z.B. der TAPS-Detektor siehe Kapitel 2.6.2).

Mit **QUEUESTATUS** wird der Zustand des Kommandopuffers ermittelt. Die Ausgabe liefert hierbei **QUEUE EMTPY** oder **COMMAND IN PROGRESS**.

Bei einer falschen Syntax des Kommandos, nicht existierenden Gerätenamen oder sonstigen Fehlern wird eine Meldung in der Form **ERROR: <FEHLERBESCHREIBUNG>** zurückgegeben.

5.6 Zugriff auf Slowcontrol-Daten

Der Zugriff auf Slowcontrol-Daten oder die Gerätesteuerung wird über das oben beschriebene Interface ermöglicht. Für den Client-Zugriff auf die Daten wurde eine C++ Klasse entwickelt, die die notwendigen Befehle an den Slowcontrol-Dämon schickt, um Daten abzurufen und Kommandos an die angeschlossenen Geräte zu senden. Die Klasse **CSlowcontrolClient** implementiert die Funktionen:

Öffentliche Funktionen der Klasse CSlowcontrolClient

```
CSlowcontrolClient();
CSlowcontrolClient(char *addr,int server_port);
~CSlowcontrolClient();

vector <int>GetData(char *device_name);
void hvon(char *device_name);
void hvoff(char *device_name);
void setvolt(char *device_name,int channel,int value);
void do_cmd(char *device_name);
```

Für die Datenakquisition wurde von der Klasse **CSlowcontrolClient** die Klasse **CSlowcontrolDAQClient** abgeleitet, die Daten aller Geräte vom Slowcontrol-Dämon abruft und in einem C-Struct ablegt. Der Abruf erfolgt über den Aufruf der Funktion *GetDAQData()*.

Öffentliche Funktionen der Klasse CSlowcontrolDAQClient

```
CSlowcontrolDAQClient();
~CSlowcontrolDAQClient();
daqdata GetDAQData();
```

Kapitel 6

Profibus und Benutzerinterface

Das Benutzerinterface und die Datenakquisition des Profibussystems sind in einem Programm zusammengefasst, welches unter Benutzung der Software LabView erstellt wurde. LabView ist ein Visualisierungssystem für Labordaten. Das System und dessen Programmiersprache G sollen hier kurz vorgestellt werden. Danach wird dann die Umsetzung der Datenanbindung an das Profibussystem und des Benutzerinterfaces der Slowcontrol beschrieben.

6.1 Programmierung in G

G ist eine graphische Programmiersprache, die es dem Anwender ermöglicht, Messwerte und Daten darzustellen. LabView bietet dabei viele Möglichkeiten, um auf externe Messinstrumente und Daten zuzugreifen. Einige der unterstützten Standards sind: ViSA, GPIB, TCP/IP, RS-232 und ActiveX.

Im Vordergrund steht die schnelle Erstellung eines Benutzerinterfaces, welches dem Anwender Daten in ansprechender Form darbietet. Zur Datenanalyse stehen einfache und komplexe Analysefunktionen zur Verfügung.

Die Programmierung des Systems erfolgt graphisch (siehe Abbildung 4.6 oder 6.1). LabView unterscheidet dabei zwischen Anzeige- und Bedienelementen, die den Datenfluss im G-Programm bestimmen. Bedienelemente und Datenerfassungselemente sind Datenquellen. Der Benutzer kann über die Verwendung von Bedienelementen Parameter und Kommandos an das LabView-Programm übergeben. Anzeigeelemente stellen Daten dar und sind somit Datenziele.

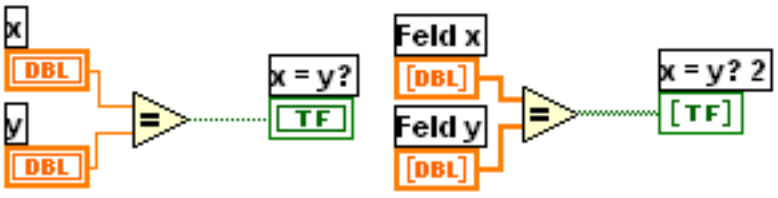

Abbildung 6.1: Vergleich von einer Zahl und einem Feld in LabView

LabView erlaubt nur die Verbindung von Datenquellen mit Datenzielen. Dazwischen können Operationen durchgeführt werden. LabView stellt neben Grundrechenarten und einfachen

Vergleichen umfangreiche Analysefunktionen zur Verfügung. Die aus anderen Programmiersprachen bekannten Datentypen können im G-Programm benutzt werden. Neben numerischen Typen, Zeichenketten, Feldern (Arrays) und Strukturen (Clustern) gibt es auch Dateihandles, Netzwerksockets und diverse andere Datentypen. Die Verwendung von Feldern und Clustern mit Daten-Operationen und Programmstrukturen ist dabei sehr intuitiv möglich. Der Vergleich zwischen zwei Feldern oder zwei numerischen Typen kann mit ein und demselben Vergleichsoperator erfolgen (siehe Abbildung 6.1). Strukturen wie for-, while- oder do-Schleifen können in den einzelnen Iterationsstufen Daten an ihrem Rand sammeln und diese beim Beenden der Struktur in Feldern zur Verfügung stellen.

Die logische Verbindung von Bedienelementen mit Operationen und Anzeigeelementen erfolgt über ein Verbindungswerkzeug.

LabView ist Multithread-fähig. Die Verarbeitung von parallelen Prozessen wird dann möglich, wenn im G-Programm Schleifen verwendet werden, die nebeneinander auf der G-Oberfläche vorhanden und nicht ineinander geschachtelt sind. Diese werden dann parallel abgearbeitet:

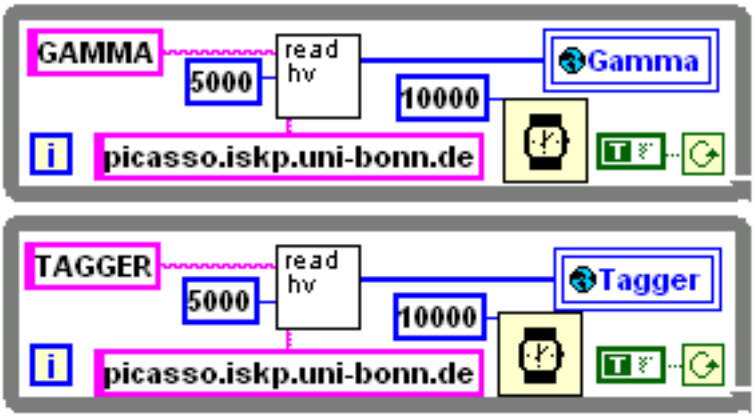

Abbildung 6.2: Parallele Ausführung von Programmteilen mit Hilfe von Schleifen

Programme werden in LabView als **Virtuelle Instrumente** (kurz: Vi) bezeichnet. Um häufig benötigte Programmteile an mehreren Stellen zu verwenden, gibt es das Konzept von **Sub-Vis**, die in anderen Programmiersprachen als Funktionen oder Prozeduren bezeichnet werden. Sub-Vis sind Virtuelle Instrumente, die über eine Schnittstelle zur Außenwelt verfügen (Terminals). Mit diesen Terminals können Daten an das Unterprogramm übergeben oder von diesem an das aufrufende Vi zurückgegeben werden. Die Terminals des Sub-Vis werden dabei mit Anzeige- und Bedienelementen auf der Benutzeroberfläche des Sub-Vis verbunden, was gleichzeitig die Richtung des Datenflusses festlegt. In Abbildung 6.2 wird gezeigt, wie über das Element *read hv* ein Sub-Vi in das G-Programm eingebunden ist. Das Sub-Vi *read hv* liest vom Slowcontrol-Dämon die Daten der Hochspannungsversorgungen und speichert sie in einer globalen Variable (Tagger, bzw. Gamma). Als Parameter werden dabei der Name der HV im Dämon, die Zieladresse und der Port für die TCP-Kommunikation übergeben. Auf diese Weise können alle HVs über das gleiche Sub-Vi abgefragt werden.

6.2 Profibus Datenerfassung

Für den Zugriff auf die Profibusgeräte stellt die Firma Beckhoff eine Funktionsbibliothek (ADS DLL[1]) zur Verfügung, die das Lesen und Setzen von Profibusvariablen ermöglicht. Die Variablen können mit Ein- und Ausgängen der Slave-Geräte verknüpft werden. Damit wird es möglich, die verschiedenen analogen und digitalen Interfaces anzusprechen. Die Konfiguration der Profibus-Variablen erfolgt über den TwinCAT System-Manager:

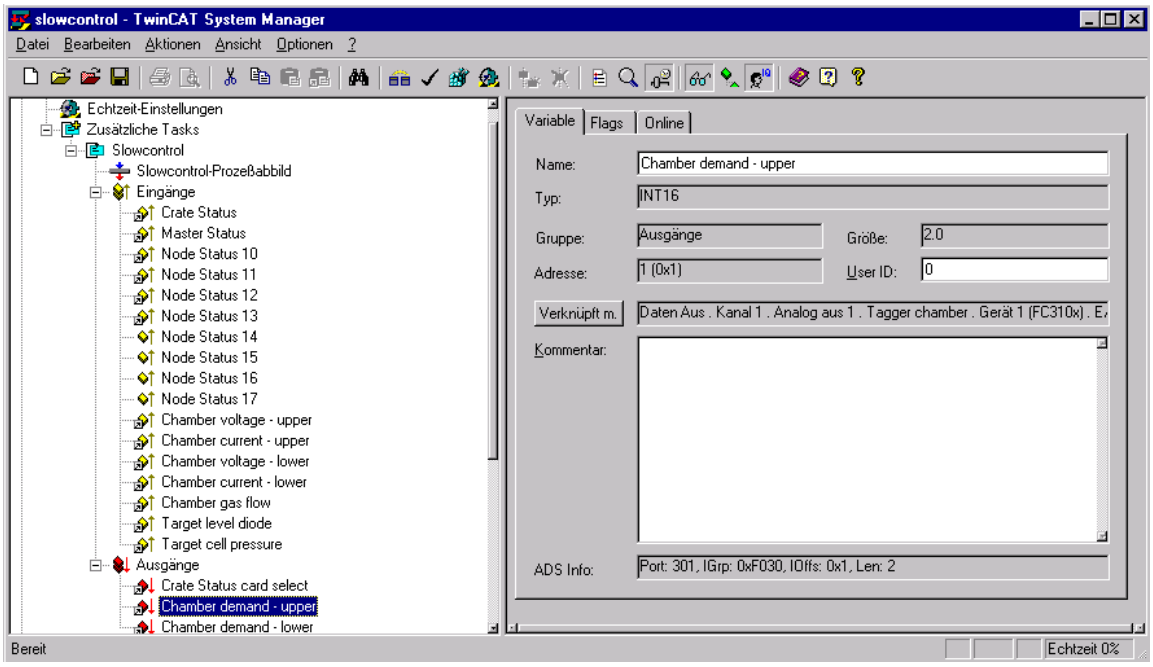

Abbildung 6.3: Konfiguration der Profibus-Variablen mit dem TwinCAT System-Manager

Jedem Eingang und Ausgang wird eine Offset-Adresse zugeteilt:

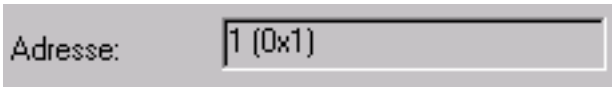

Abbildung 6.4: Adressregister im Profibus-System

Für die Ein- und Ausgänge sind zwei getrennte Adressbereiche festgelegt. Die Unterscheidung erfolgt über die Indexgruppe, die für die Ausgänge **0xF030**[2] und für die Eingänge **0xF020** festgelegt wurde. Um auf Register zugreifen zu können, benötigt man:

- den ADS-Port des Systems
- die Indexgruppe
- den Offset

[1] Dynamic Link Library (DLL)
[2] hexadezimal

- den Datentyp

Der ADS Port des TwinCat I/O Systems ist **301**. Der Offset gibt die Position einer Variablen im Ein- oder Ausgangsregister an, die mit dem Systemmanager für jede Variable festgelegt wird. Eine Aufstellung über die in der Slowcontrol verwendeten Register kann dem Anhang C entnommen werden.

Abbildung 6.5: ADS-Informationen im TwinCAT System-Manager

Die mitgelieferte Bibliothek erlaubt den Zugriff auf diese Variablen. Um den Zugriff aus dem LabView System zu vereinfachen, wurde eine weitere Bibliothek entworfen, die als Interface zwischen der Beckhoff-Bibliothek und dem G-Programm dient. Die *profibus.dll* stellt den Lese- und Schreibzugriff auf die Datentypen byte, word und short integer zur Verfügung:

profibus.dll Funktionen

```
void OpenLocalPort(void);
void SetPort(int port);
CloseLocalPort(void);

unsigned char read_byte(unsigned int offset);
unsigned int read_word(unsigned int offset);
signed int read_int(unsigned int offset);

write_byte(unsigned int offset,unsigned char data);
write_word(unsigned int offset,unsigned int data);
write_int(unsigned int offset,signed int data);
```

Die Funktionen *read_byte()*, *read_word()* und *read_int()* bedienen sich dabei der Funktion *AdsSyncReadReq()* der Beckhoff-Bibliothek und verwenden als Indexgruppe **0xF020**, so dass synchron lesend auf die Eingangsregister zugegriffen wird. Die write-Funktionen schreiben in die Ausgangsregister mit der Indexgruppe **0xF030** und der Funktion *AdsSyncWriteReq()*. Die Funktionen *OpenLocalPort()* und *CloseLocalPort()* öffnen und schließen einen Kommunikationsport für die ADS Kommunikation. Die Portadresse für die Kommunikation wird über die Funktion *SetPort()* festgelegt. Der Zugriff auf das Profibussystem geschieht dann folgendermaßen:

Beispiel eines Datenzugriffs

```
OpenLocalPort(void);
SetPort(301);
  Aufruf von Lese- und Schreibfunktionen
CloseLocalPort(void);
```

Diese Funktionen wurden in einem Sub-Vi zusammengefasst, um den leichteren Zugriff aus dem LabView-Programm zu ermöglichen. Die Datei **profibus_write_byte.vi** enthält das Sub-Vi, welchem der Kommunikationsport, die Offset-Adresse und der zu schreibenden Byte-Wert über Terminals übergeben werden. Der übergebene Byte-Wert wird in die Profibus-Ausgangsregister geschrieben. Hier ist die Sequenz dargestellt, die nacheinander die oben beschriebenen Funktionsaufrufe über externe DLL-Aufrufe durchführt:

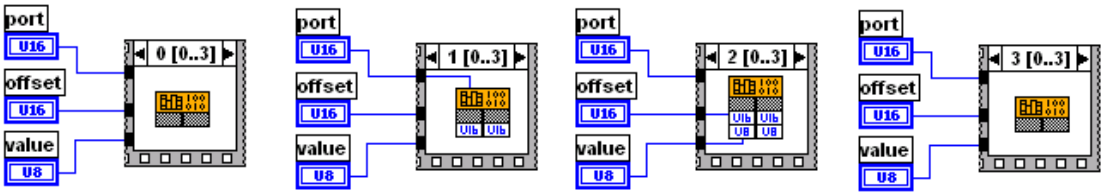

Abbildung 6.6: Sub-Vi *profibus_write_byte* zur Ansteuerung des Profibus-Systems

Neben diesem Sub-Vi existieren für die Datentypen byte, word und integer ähnliche Sub-Vis. Die vom Profibus zur Verfügung gestellten Daten werden nun über **profibus_cycle.vi** erfasst und in einer globalen Variable (in **global_var.vi**) gespeichert. Dazu gehören die Daten

- des Targets
- der Taggerkammern (Gasfluss, Spannungen und Ströme)
- des Crate-Kontrollsystems
- und der Status des Profibus-Systems

Die Speicherung der Daten in einer globalen Variable ermöglicht es, unabhängig von der Datenerfassung auf die Profibus-Daten zuzugreifen.

6.3 Übergabe der Profibus-Daten an den Slowcontrol-Dämon

Das Sub-Vi *slowctrld_transfer_data.vi* ist für die Datenübergabe der Profibus-Daten an den Slowcontrol-Dämon zuständig. Der Zugriff auf die Daten erfolgt über die globale Variable **Profibus Data**. Sie ist ein aus vier numerischen Feldern zusammengesetzter Cluster[3], welcher die in Kapitel 6.2 angegebenen Daten enthält. Unter Benutzung der Clusterfunktionen wird der Cluster in einzelne Felder aufgetrennt. Jedes numerische Feld wird dann dem Sub-Vi über ein Terminal übergeben und von diesem in eine Zeichenkette umgewandelt. Unter Verwendung des **STORE** Befehls werden die Daten im Slowcontrol-Dämon gespeichert. Die Verbindung zum Slowcontrol-Dämon wird über das TCP Protokoll aufgebaut. Die Adresse und der Port für die Kommunikation werden dem Vi *slowctrld_transfer_data.vi* von außen über die Terminals übergeben. Die Datenübergabe an den Slowcontrol-Dämon unter Benutzung des Sub-Vis gestaltet sich dann in der Programmiersprache G wie folgt:

Jeder Instanz des Sub-Vis wird eine Bezeichnung in Form einer Zeichenkette zugewiesen, die als Referenz im Slowcontrol-Dämon dient. Mit dieser Referenz wird später auf die Daten zugegriffen (siehe Kapitel 5.5). Die Datenübertragung wird einmal pro Minute ausgeführt.

[3] Zusammenfassung von mehreren Datentypen zu einer Variablen - in C/C++ wird dies als struct bezeichnet

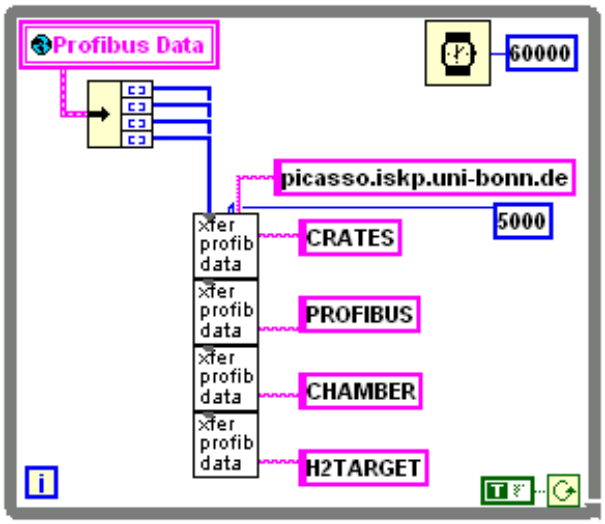

Abbildung 6.7: Übergabe der Slowcontrol-Daten an den Dämon

6.4 Das Benutzerinterface

Das Benutzerinterface besteht aus abgetrennten Bereichen, welche jeweils die Benutzerschnittstelle der zu überwachenden oder zu steuernden Komponente enthalten. Die Komponenten des implementierten Interfaces sollen hier kurz vorgestellt werden. Es soll nur die Funktionalität beschrieben werden, ohne auf die Programmierung einzugehen.

6.4.1 Slowcontrol-Überwachung

Die Slowcontrol verfügt über ein Überwachungssystem für ihre Datenakquisition. Hierbei werden die Master-Station, die Slaves des Profibussystems und die Funktion des Slowcontrol-Dämon kontrolliert.

Um die Funktion des Dämon-Prozesses zu verifizieren, wird eine TCP-Verbindung vom Slowcontrol-PC zum Server des Dämons aufgebaut. Kann eine Verbindung hergestellt werden, so ist der Abruf der Daten sichergestellt. Dieser Status wird in die globale Variable **Slowcontroldaemon Status** geschrieben (boolescher Wert). Sollte der Dämon durch eine Unterbrechung der Netzwerkverbindung, durch Beenden des Prozesses oder Abschalten des Servers, auf dem der Dämon gestartet wurde, nicht mehr erreichbar sein, so werden alle Benutzerinterfaces, die den Slowcontrol-Dämon als Datenquelle benötigen, deaktiviert, so dass der Benutzer keinen Zugriff mehr auf diese erhält. Der Status des Slowcontrol-Dämons wird auf der Benutzeroberfläche durch eine zweifarbige Leuchte[4] angezeigt. Der Dämon wird auf dem Linux-Server in die Datei */etc/inittab* mit der Option *respawn* eingetragen. Dadurch wird ein automatischer Neustart des Dämon-Prozesses veranlasst.

Die Funktion des Profibussystems wird anhand von Statusregistern der Profibus-Slaves und dem Register des Masters überwacht (siehe Anhang C). Mit dem Vi *profibus_node_status_all_nodes.vi* werden alle Register überprüft. Konnten keinerlei Fehler festgestellt werden, so leuchtet die dazugehörige Statusanzeige grün.

[4] grün: Verbindung konnte aufgebaut werden - rot: keine Verbindung möglich

Für den Gesamtstatus der Slowcontrol-Datenerfassung wird ein logisches UND aus allen Statusinformationen gebildet und auf der Benutzeroberfläche angezeigt:

Abbildung 6.8: Status der Slowcontrol

6.4.2 Beam-Scan

Das hier gezeigte Interface des Beam-Scanners zeigt ein Profil des Photonenstrahls, wie es während einer Strahlzeit im März 2001 aufgenommen wurde. Die beiden Maxima liegen sehr genau auf den markierten Sollpositionen, und das Profil zeigt einen glatten Verlauf. Für den Experimentbetrieb wird ein solches γ-Strahlprofil benötigt.

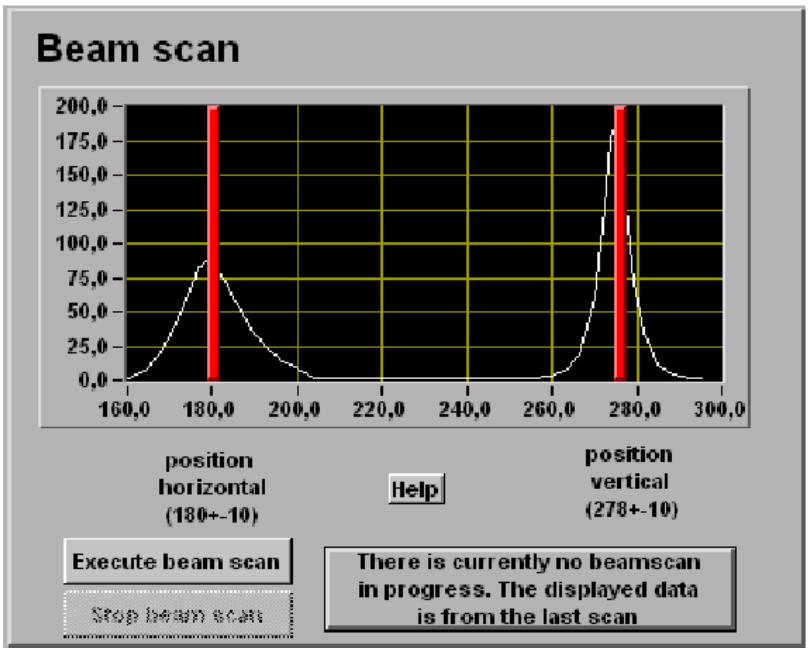

Abbildung 6.9: Benutzerinterface des Beam-Scanners

Der Benutzer kann über das Benutzerinterface einen Scan starten bzw. einen laufenden stoppen. Dies bietet die Möglichkeit, während der Einstellung der Beamline des Experiments die Qualität des Photonenstrahls zu ermitteln. Ohne Eingriff durch den Benutzer wird alle

zwei Stunden automatisch ein Beam-Scan ausgeführt. Die Daten des Beam-Scans können außerdem noch mit einem externen Programm dargestellt werden, welches versucht, an die Strahlpositionen eine Gauss-Funktion zu fitten. Die Daten der Beam-Scans werden nicht nur im Rahmen der DAQ gespeichert, sondern zusätzlich in einzelnen Dateien festgehalten.

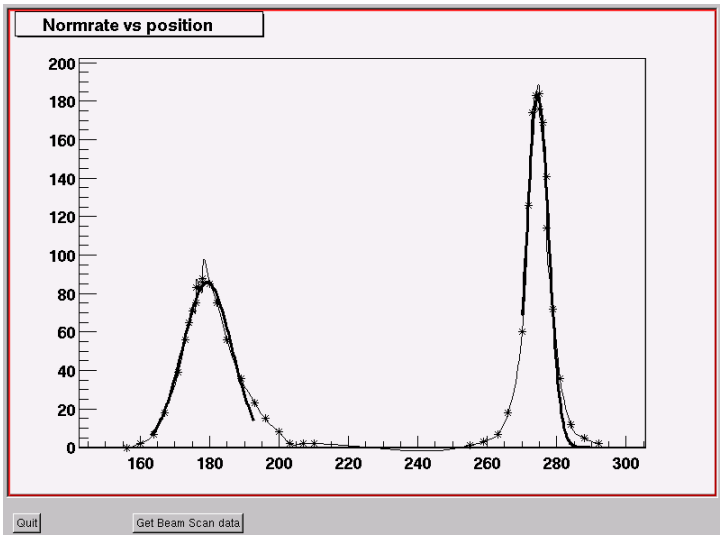

Abbildung 6.10: Externes Anzeigeprogramm für den Beam-Scanner

6.4.3 HV-Steuerung

Die Anzeige der HV-Steuerung zeigt den Status aller Hochspannungen an. Falls ein Fehler vorliegt, so wechselt ein Anzeigeelement von grün auf rot. Ein Fehler wird dabei entweder durch die Fehlerbehandlungsroutinen der Hochspannungsversorgungen ermittelt oder über eine Abweichung zwischen Soll- und Ist-Spannungswerten.

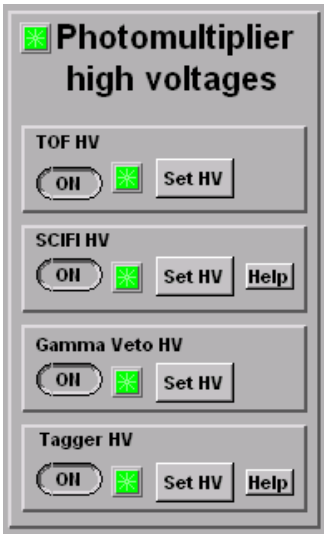

Abbildung 6.11: Status aller Hochspannungsversorgungen

Über das Bedienelement **Set HV** kann ein Kontrollinterface für jede der Hochspannungsversorgungen aufgerufen werden. Damit ist es möglich,

- die Hochspannung ein- und auszuschalten,
- Kanäle auf neue Spannungen zu setzen,
- Kanäle auf Fehler zu überprüfen,
- Spannungskonfigurationen zu laden und zu speichern.

Jeder HV-Typ verfügt dabei über ein eigenes Interface. Das Interface der Hochspannungsversorgung der Szintillatoren des Taggingsystems sieht folgendermaßen aus:

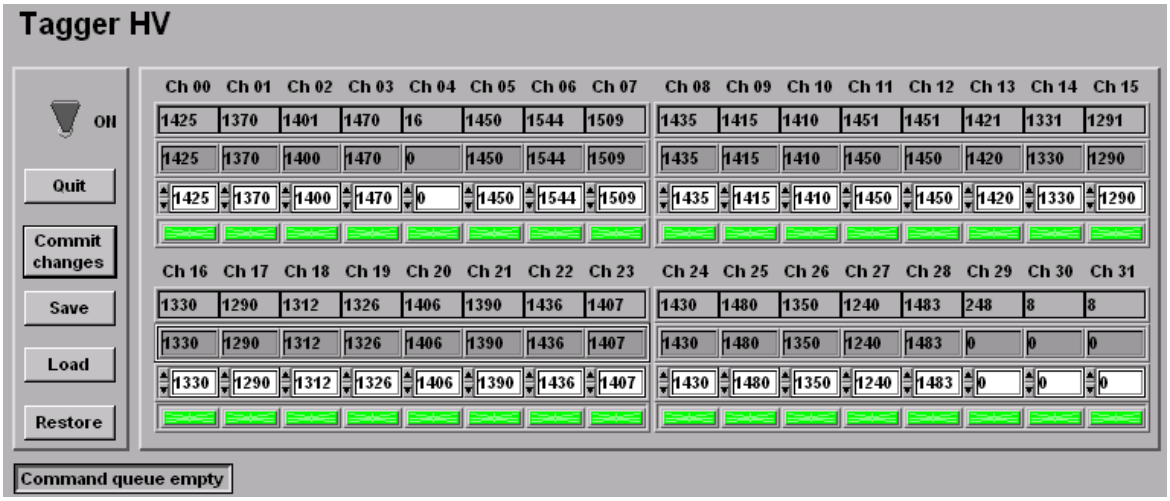

Abbildung 6.12: Benutzerinterface der Tagger-HV (ohne Hilfe)

Die Anzeige enthält die Soll- und Ist-Werte der Kanäle der Hochspannungsversorgungen und eine Möglichkeit zur Einstellung neuer Sollspannungen. Jeder Kanal verfügt außerdem noch über eine Statusanzeige. Nach Eingabe der neuen Sollspannungen können mit der Taste *Commit changes* die entsprechenden Befehle an den Slowcontrol-Dämon gesendet werden. Diese werden an das Gerät übermittelt, wenn der laufende Befehl zur Bestimmung der Soll- oder Ist-Spannung beendet wurde. Das Benutzerinterface wird sofort gesperrt, bis alle Befehle an die HV übermittelt wurden.

6.4.4 Crate-Kontrolle

Für die Überwachung der Detektorelektronik wird das am CERN entwickelte Cratekontrollsystem (Kapitel 3.1.2) weiterverwendet, welches über insgesamt 64 optische Anschlüsse verfügt. Je acht Anschlüsse sind auf einer Karte untergebracht. Über einen Multiplexer werden die acht Karten mit einem 3-bit[5] Eingang angesprochen.

Der 8-bit-Ausgang des Kontrollsystems wurde an digitale Eingänge des Profibus-Systems angeschlossen. Ein digitaler Ausgang des Profibus-Systems steuert den Multiplexer des Cratecontrollers.

[5] mögliche Dezimalzahlen 0-7

Jeder der 64 Anschlüsse des Cratecontrollers besteht aus einem optischen Empfänger. Die Überwachungsmodule der Crates liefern über einen optischen Sender SFH-450V/SFH-750V der Firma Siemens/Infineon [21] ein Signal, wenn das jeweilige Elektronikcrate normal funktioniert. Bei Fehlfunktionen der Spannungsversorgungen, der Kühlung oder einem kompletten Ausfall wird kein Signal gesendet.

Module waren noch vom CB-LEAR-Experiment für die NIM-, Camac-, Shaper-Crates und die Vorverstärker vorhanden. Im Rahmen dieser Arbeit wurden Module für die Fastbus- und VME-Crates entwickelt. Die Fastbus-Crates verfügen über eine 37-polige Sub-D Monitorbuchse an der Frontseite, an der Statusinformationen für das Crate abgegriffen werden können. Im Crate wird dabei ein Relais geschlossen, das die Pins 2 und 3 miteinander verbindet, wenn keinerlei Fehlfunktion vorliegt. Im Falle einer Fehlfunktion wechselt der Ausgang vom Zustand 0Ω in einen hochohmigen Zustand. Mit einer Spannung von 5 Volt, die ebenfalls am Monitorausgang anliegt, wird eine Spannung für den optischen Sender generiert.

Für die VME-Crates wurde eine kleine Platine aufgebaut, die über eine Komparatorschaltung (MAX 8213 IC) ein optisches Signal erzeugt, wenn die Spannungen des Crates innerhalb von 5 ± 0.5 Volt, bzw. 12 ± 1 Volt und -12 ± 1 Volt liegen. Die Schaltung und der Aufbau der Platine kann dem Anhang D entnommen werden.

Um alle Anschlüsse des Cratekontrollsystems ansprechen zu können, muss der Ausgang des Multiplexers alle möglichen Binärkombinationen der 3 Bits durchlaufen. Nach dem Setzen des jeweiligen Bitmusters muss das Profibussystem zunächst die neuen Ausgangsregister an den Profibus-Slave übertragen, um danach die neuen Werte der Eingangsregister lesen zu können. Zwischen dem Setzen der Ausgangsregister und dem Einlesen der neuen Eingangsregister muss eine Zeitdauer von mindestens 20 ms abgewartet werden, damit das Profibussystem die Daten übertragen kann.

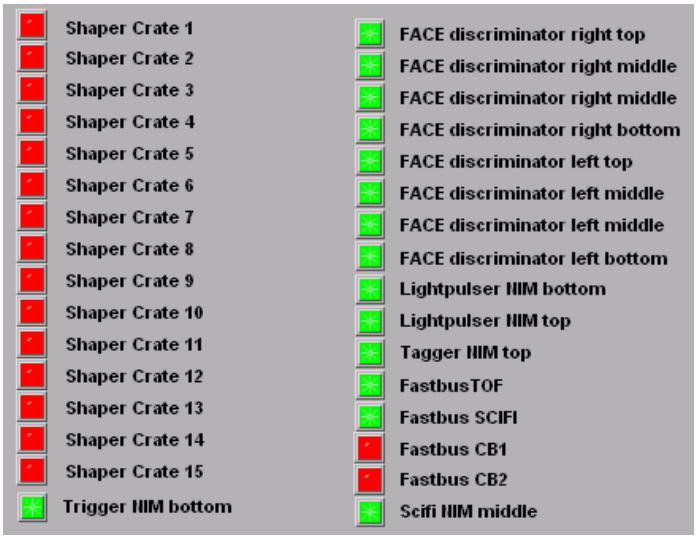

Abbildung 6.13: Status der Elektronikcrates

Das Vi *profibus_read_crate_controller.vi* setzt das Ausgangsregister für die Ansteuerung des Multiplexers auf den vom Terminal übergebenen Wert. *profibus_crate_status.vi* bedient sich des Vi *profibus_read_crate_controller.vi* und liest alle 8 Kanäle des Multiplexers aus, um diese in einer globalen Variablen zu speichern. Im Slowcontrol-Programm wird mit dieser Information der Gesamtstatus der Elektronikcrates ermittelt und angezeigt. Bei einem Ausfall

einzelner Elektronikcrates kann die Darstellung erweitert werden. Die in Abbildung 6.13 gezeigte erweiterte Anzeige zeigt einen Teil der überwachten Elektronikcrates.

6.4.5 Taggerkammern

Die Überwachung und Steuerung der Taggerkammern beinhaltet:

- Messung der Spannung der Drahtkammern
- Messung des Stromflusses
- Einstellung der Hochspannung
- Messung des Gasflusses durch die Kammern
- Abschaltung der Kammern bei zu hohem Stromfluss

Der Gasfluss wird mit Hilfe eines Messgerätes bestimmt, welches ein analoges Spannungssignal liefert. Dieses Spannungssignal wird über eine analoge Eingangsklemme des Profibussystems erfasst und auf einer Anzeige dargestellt.

Die Spannungen der Drahtkammern können an der Hochspannungsversorgung mit einer Übersetzung von 1/1000 gemessen bzw. eingestellt werden. Die Vorgabe einer Spannung erfolgt mit zwei analogen Ausgängen getrennt für beide Kammern. Die Spannungen und Ströme werden mit insgesamt vier analogen Eingängen des Profibussystems gemessen und dargestellt.

Im G-Programm kann für die beiden Kammern ein maximaler Strom definiert werden. Überschreitet der Stromfluss in einer der beiden Kammern diesen Wert, so veranlasst die Slowcontrol, dass die Hochspannung für die jeweilige Drahtkammer sofort auf 0 Volt heruntergeregelt und ein Alarm ausgelöst wird.

6.4.6 Sonstige Kontrollen

Neben den genannten Kontrollen sind im Slowcontrol-System noch Anzeigen für den Status des Flüssigwasserstoff-Targets und die Magnetfeldstärke des Tagging-Magneten vorhanden. Die Anzeigen stellen die Daten, die vom Profibus-System bzw. dem Slowcontrol-Dämon gewonnen wurden, graphisch dar. Über- bzw. unterschreiten die gemessenen Werte einen im G-Programm einstellbaren Grenzwert, so wird die Alarmroutine ausgelöst und der Benutzer optisch und akustisch über den Fehler informiert.

6.4.7 Alarmsystem

Jede Komponente des Benutzerinterface enthält neben der Anzeige der jeweiligen Daten eine Gesamtstatusanzeige. Alle Statusinformationen der Komponenten werden logisch miteinander verknüpft und als Alarmstatus auf der Benutzeroberfläche dargestellt. Werden Grenzwerte überschritten oder fallen überwachte Komponenten aus, so wird dies auf der Benutzeroberfläche angezeigt. Um Fehlalarm zu vermeiden, wird bei jedem Alarm eine Fehlerbehandlungsroutine ausgelöst, die zunächst überprüft, ob innerhalb von 10 Sekunden der

Fehler weiterhin auftritt. Ist dies der Fall, so wird ein akustischer Alarm im Kontrollraum des Experiments ausgelöst. Der Benutzer hat dann die Möglichkeit, auf den Fehler zu reagieren und ihn zu beheben. Außerdem kann der akustische Alarm für eine bestimmte Zeitdauer unterbunden werden.

Der akustische Alarm wird auf den am Kontrollrechner des Experiments angeschlossenen Lautsprechern ausgegeben. Durch den Aufbau einer TCP-Verbindung zu Port 5001 auf dem Kontrollrechner wird der Alarm gestartet und kann über eine TCP-Verbindung zu Port 5002 wieder gestoppt werden. Realisiert wurde dies über Linux Shell-Scripts, die durch den **inetd**[6] gestartet werden. In der Konfigurationsdatei */etc/inetd.conf* sind die beiden Skripte wie folgt eingetragen:

/etc/inetd.conf Einträge für den Slowcontrol-Alarm

```
slowalert stream tcp nowait root /home/cb/SlowCtrl/alert/slowalert
slowalertstop stream tcp nowait root /home/cb/SlowCtrl/alert/slowalertstop
```

slowalert und slowalertstop sind in der Datei */etc/services* definiert und auf die Ports 5001 und 5002 gesetzt.

6.4.8 Hilfesystem

Bei einem Ausfall ist es wichtig, dass der jeweilige Benutzer Informationen und Fehlerbeschreibungen erhält, so dass er die aufgetretenen Fehler wenn möglich selbstständig beheben kann. Zu diesem Zweck wurde ein kleines Hilfesystem implementiert, das eine Anleitung zur Vorgehensweise bei Fehlern bereithält. Durch Anwählen von kleinen **Help**-Elementen auf der Oberfläche kann der Benutzer diese Informationen abrufen.

Neben diesen Knöpfen sind noch **Locate**-Knöpfe vorhanden, mit Hilfe derer Positionen von Komponenten graphisch auf einem Plan des Experimentaufbaus angezeigt werden können, um somit fehlerhafte Komponenten leicht räumlich lokalisieren zu können.

[6] Unix Programm zur Ausführung von konsolenorientierten Dämonen

Kapitel 7

Zusammenfassung und Ausblick

Das in dieser Diplomarbeit beschriebene Überwachungs- und Kontrollsystem wurde im Rahmen dieser Arbeit geplant, entwickelt und aufgebaut. Neben der Programmierung der Software mit LabView und der Programmiersprache C/C++ wurde die notwendige Hardware entworfen, aufgebaut und installiert. Das Profibus-System wurde beschafft und in der PHOENICS-Halle und der SAPHIR-Area in Betrieb genommen. Lediglich der Multiplexer und einige Module des Cratekontrollsystems wurden vom CB-LEAR-Experiment unverändert übernommen.

Das so implementierte Slowcontrol-System hat schon während des Aufbaus und darüber hinaus wertvolle Dienste geleistet. Der Experimentbetrieb muss bei Änderungen z.B. der Hochspannungen der verschiedenen Photomultiplier nicht mehr unterbrochen werden. So sind Einstellungen wesentlich schneller und unkomplizierter vorzunehmen.

Die Überwachung der Detektorkomponenten und der Elektronik, sowie die Notabschaltung der Drahtkammern des Taggingsystems bei zu hohen Strömen, funktionieren einwandfrei. Insbesondere durch die Abschaltung der Drahtkammern konnten schon Beschädigungen vermieden werden. Auch Fehler einzelner Kanäle der Hochspannungsversorgungen wurden korrekt erkannt und an den Benutzer gemeldet, so dass die Datennahme relativ zügig wieder aufgenommen werden konnte, nachdem die Fehler behoben waren. Die Überwachung der Elektronik-Crates hat den Ausfall von Elektronik-Komponenten des Fast-Cluster-Encoders erkannt und somit verhindert, dass Daten aufgezeichnet wurden, die keine sinvolle Auswertung zulassen.

Der Aufbau des hier implementierten Systems ist modular ausgelegt, so dass es keine Schwierigkeiten bereiten wird, weitere Detektorkomponenten einzufügen. Hier sei insbesondere der TAPS-Detektor erwähnt, aber auch die Temperatursensoren des Crystal-Barrel-Kalorimeters, die zum gegenwärtigen Zeitpunkt noch nicht angeschlossen werden konnten, da die Zuleitungen noch nicht nach außen geführt sind. Dies sollte während der Umbauphase für das TAPS-Detektorsystem geschehen, so dass diese ebenfalls mit überwacht werden können. Änderungen an Parametern und an der Benutzeroberfläche sollten nach einer kurzen Einarbeitungszeit jedem Benutzer möglich sein.

Das Slowcontrol-System wird während des fortlaufenden Betriebs hoffentlich gute Dienste leisten und dafür sorgen, dass Fehler frühzeitig entdeckt, bzw. dass Beschädigungen ganz oder teilweise vermieden werden und so zum Erfolg des Crystal-Barrel-Experiments an der Elektronen-Stretcher-Anlage ELSA beitragen.

Anhang A

RS-232/TTY-Befehlssequenzen

Für die Ansteuerung der verschiedenen Geräte über RS-232/TTY werden unterschiedliche Befehle für die einzelnen Gerätetypen benötigt. Hier folgt eine Aufstellung der im Slowcontrol-Dämon (siehe Kapitel 5) verwendeten Befehlssequenzen. <CR> ist hierbei das Zeichen Carriage-Return[1] und <LF> das Zeichen für Linefeed[2].

Scaler SC 8000 - Beam-Scanner

Befehl	Funktion
R<CR><LF>	Lesen der Einstellungen des Displays
C<CR><LF>	Display zurücksetzen (Zähler auf 0)
A<CR><LF>	Display lesen

Isel IT 116 Schrittmotorsteuerung - Beam-Scanner

Befehl	Funktion
@01<CR><LF>	Anzahl Achsen für Gerät 0 auf 1 setzen
@0R1<CR><LF>	Gerät 0 auf Referenzposition fahren
@0i1<CR><LF>	Achse 1 auf Gerät 0 anwählen
@0A sh,sp<CR><LF>	Bewegung um sh Einheiten mit Geschwindigkeit sp

Le Croy HV4032A - Gamma-Veto und Tagger

Jedem Befehl ist jeweils die an der HV eingestellte Mainframeadresse voran zu stellen. M16N bedeutet: einschalten der Hochspannung für die Hochspannungsversorgung mit der Mainframeadresse 16.

Befehl	Funktion
N<CR>	Einschalten
F<CR>	Ausschalten
S<CR>	Status abfragen
V<CR>	aktuelle Spannungen auslesen
D<CR>	gesetzte Spannungen auslesen
C<Nummer><Spannung><CR>	Kanal <Nummer> auf <Spannung> Volt setzen

[1] Eingabetaste
[2] Zeilenvorschub

Le Croy HV1445 - TOF

Jedem Befehl ist jeweils die an der HV eingestellte Mainframeadresse voran zu stellen. M8S bedeutet: Abrufen des Status für die HV mit der Mainframeadresse 8.

Befehl	Funktion
N<CR>	Einschalten
F<CR>	Ausschalten
S<CR>	Status abfragen
V<CR>	aktuelle Spannungen auslesen
D<CR>	gesetzte Spannungen auslesen
C<Nummer><Spannung><CR>	Kanal <Nummer> auf <Spannung> Volt setzen

Le Croy HV1450 - Innendetektor

Befehl	Funktion
1450<CR>	Login in das Benutzerinterface
QUIT<CR>	Ausloggen aus dem Benutzerinterface
HVON<CR>	Einschalten
HVOFF<CR>	Ausschalten
HVSTATUS<CR>	Status abrufen
RC L<Modul> MV	aktuelle Spannungen aus <Modul> abrufen
RC L<Modul> DV	gesetzte Spannungen aus <Modul> abrufen
LD L<Modul>.<Kanal> DV <Spannung>	<Modul> und <Kanal> auf <Spannung> setzen

MPS FH54 - Hall-Sonde Taggingmagnet

Befehl	Funktion
?TEMP	Abfrage der Temperatur in °C
?MEAS	Abfrage des Magnetfeldes in Tesla oder mTesla

Anhang B

Slowcontrol-Bank (RSLC)

bank size in words (16 bit) incl. header			
Crate Status (Crate 0-15)		Crate Status (Crate 16-31)	
Crate Status (Crate 32-47)		Crate Status (Crate 48-63)	
RESERVED			
RESERVED			
Profibus Node 0	Profibus Node 1	Profibus Node 2	Profibus Node 3
...	...	...	...
Profibus Node 28	Profibus Node 29	Profibus Node 30	Profibus Node 31
Target level diode		Target cell pressure	
RESERVED			
RESERVED			
RESERVED			
Tagger field (mT)		RESERVED	
Tagger temperature ($10^{-}1C$		RESERVED	
Tagger chamber flow		RESERVED	
RESERVED		RESERVED	
Beam scan position 0		Beam scan rate 0	
...		...	
Beam scan position 199		Beam scan rate 199	
Gamma Veto HV current volt Ch 0		Gamma Veto HV demand volt Ch 0	
Gamma Veto HV status Ch 0		Reserved Ch 0	
...		...	
Gamma Veto HV current volt Ch 31		Gamma Veto HV demand volt Ch 31	
Gamma Veto HV status Ch 31		Reserved Ch 31	
Tagger HV current volt Ch 0		Tagger HV demand volt Ch 0	
Tagger HV status Ch 0		Reserved Ch 0	
...		...	
Tagger HV current volt Ch 31		Tagger HV demand volt Ch 31	
Tagger HV status Ch 31		Reserved Ch 31	
0 ... 15		16 ... 31	

TOF HV current volt Ch 1	TOF HV demand volt Ch 1
TOF HV status Ch 1	Reserved Ch 1
...	...
TOF HV current volt Ch 255	TOF HV demand volt Ch 255
TOF HV status Ch 255	Reserved Ch 255
Scifi HV current volt Ch 1	Scifi HV demand volt Ch 1
Scifi HV status Ch 1	Reserved Ch 1
...	...
Scifi HV current volt Ch 40	Scifi HV demand volt Ch 40
Scifi HV status Ch 40	Reserved Ch 40
Tagger top chamber HV current volt	Tagger top chamber HV demand volt
Tagger top chamber HV current	Reserved top chamber
Tagger bottom chamber HV current volt	Tagger bottom chamber HV demand volt
Tagger bottom chamber HV current	Reserved bottom chamber
0 ... 15	16 ... 31

Anhang C

Register im TwinCAT System-Manager

Eingänge - ADS Port 301 - Indexgruppe 0xF020

Offset	Variable	Typ	Funktion
0x00 - 0	Crate status	byte	8-bit Ausgang des Cratecontrollers
0x0A - 10	Master status	byte	Status Profibus Master
0x0B - 11	Node Status 10	byte	Status Profibus Slave ID 10
0x0C - 12	Node Status 11	byte	Status Profibus Slave ID 11
0x0D - 13	Node Status 12	byte	Status Profibus Slave ID 12
0x0E - 14	Node Status 13	byte	Status Profibus Slave ID 13
0x0F - 15	Node Status 14	byte	Status Profibus Slave ID 14
0x10 - 16	Node Status 15	byte	Status Profibus Slave ID 15
0x11 - 17	Node Status 16	byte	Status Profibus Slave ID 16
0x12 - 18	Node Status 17	byte	Status Profibus Slave ID 17
0x32 - 50	Chamber voltage - upper	short integer	obere Taggerkammer Spannung
0x34 - 52	Chamber current - upper	short integer	obere Taggerkammer Strom
0x36 - 50	Chamber voltage - lower	short integer	untere Taggerkammer Spannung
0x38 - 52	Chamber current - lower	short integer	untere Taggerkammer Strom
0x3A - 54	Chamber gas flow	short integer	Gasfluss durch beide Kammern
0x46 - 70	Target level diode	short integer	Leveldiode im Target
0x48 - 72	Target cell pressure	short integer	Zellendruck der Targetzelle

Ausgänge - ADS Port 301 - Indexgruppe 0xF030

Offset	Variable	Typ	Funktion
0x00 - 0	Crate status card select	byte	Auswahl Karte Cratecontroller
0x01 - 1	Chamber demand - upper	short integer	obere Taggerkammer Einstellung Spannung
0x03 - 3	Chamber demand - lower	short integer	untere Taggerkammer Einstellung Spannung

Anhang D

VME-Spannungsüberwachung

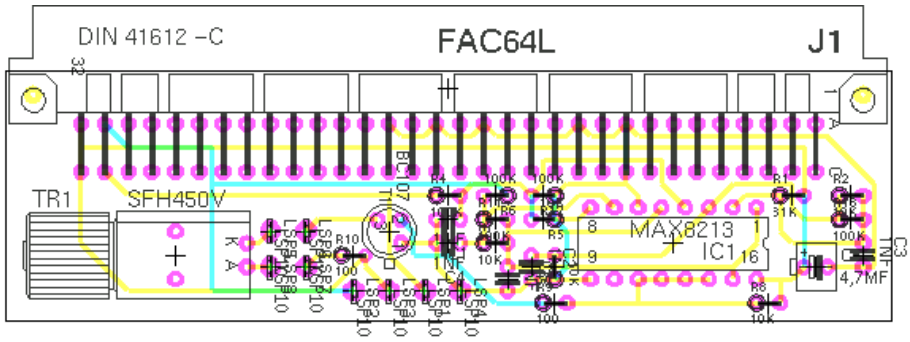

Abbildung D.1: VME-Spannungsüberwachungsplatine

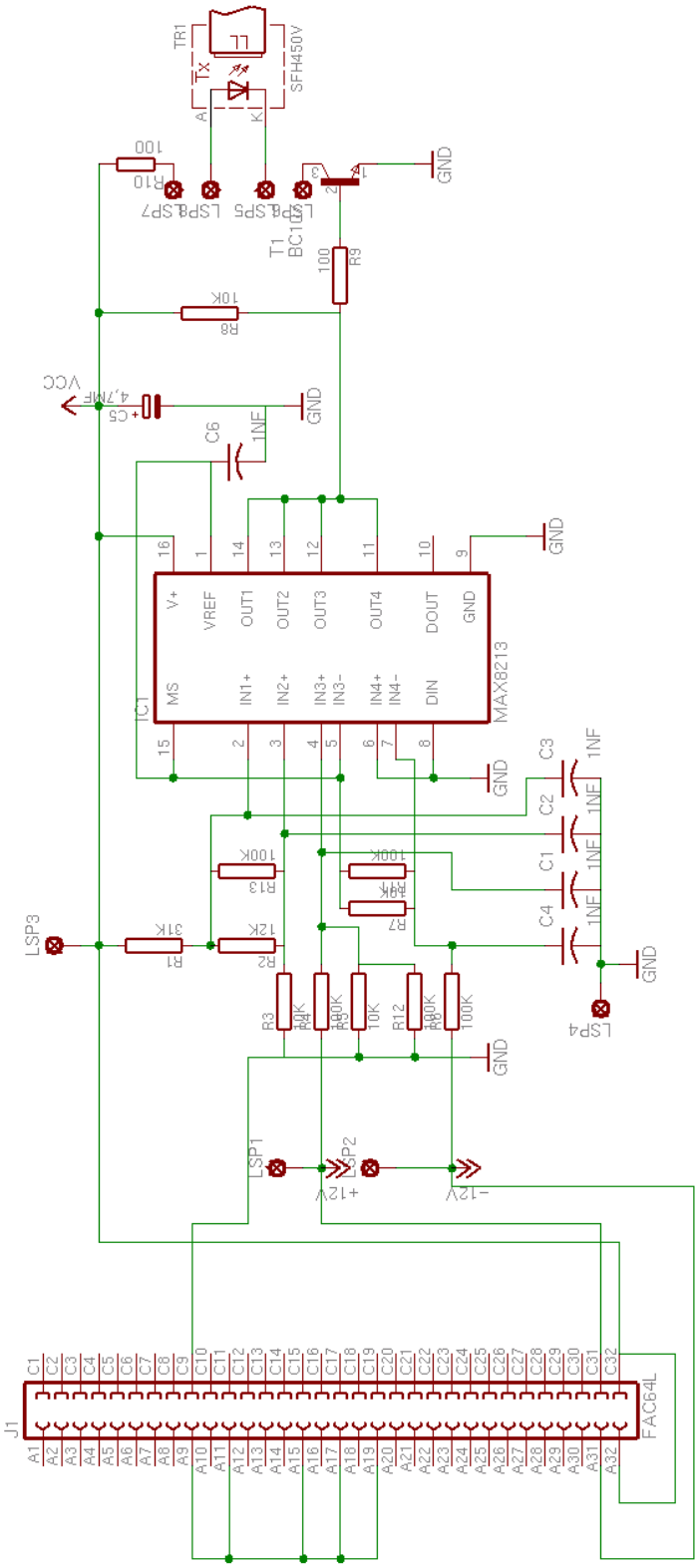

Abbildung D.2: Schaltplan VME-Spannungsüberwachung

Anhang E

Übersicht über die LabView-Vis

Dateiname	Funktion
8_bool_array_to_8_bool.vi	8-faches boolesches Feld aufteilen
8_byte_array_to_8_8_boolean.vi	8-faches Byte Feld bitweise in 8x8 boolesches Feld
alert_handler.vi	Alarmbehandlung
get_8x_subset.vi	8-faches Feld in 8 Zahlen aufspalten
global_var.vi	globale Variablen
led_red_green.ctl	Bedienelement Anzeige rot-grün
slowcontrol.vi	Hauptprogramm

Tabelle E.1: Vis des Hauptprogramms und Hilfs-Vis

Dateiname	Funktion
display_crate_status_single.vi	Fenster zur Anzeige der überwachten Crates
display_crate_status_single_all_channels.vi	zeigt alle Kanäle des Cratecontrollers
display_node_status_single.vi	Fenster zur Anzeige einzelner Profibus-Nodes
hv_tagger_chamber_check_current.vi	überprüft Strom der Taggerkammern
hv_tagger_chamber_check_failure.vi	überprüft Kammer-Spannungen und Ströme
hv_tagger_chamber_get_voltage_and_current.vi	ermittelt Kammer-Spannungen und Ströme
hv_tagger_chamber_on.vi	überprüft, ob Kammern eingeschaltet sind
hv_tagger_chamber_read_demand.vi	liest gesetzte Spannungen
hv_tagger_chamber_set_voltage.vi	setzt neue Spannungen für die Kammern
tagger_chamber_flow.vi	ermittelt und überpüft den Gasfluss
target_check.vi	überprüft H2-Targetparameter
target_data.vi	ermittelt Targetdaten

Tabelle E.2: Vis für Geräte am Profibus

Dateiname	Funktion
profibus_crate_status.vi	Abfrage des Cratekontroll-Systems
profibus_crate_status_all_crates.vi	überprüft den Zustand aller Crates
profibus_cycle.vi	Profibusdatenabfrage-Zyklus
profibus_node_status.vi	Abfrage des Status der Profibus-Nodes
profibus_node_status_all_nodes.vi	erzeugt boolesche Werte aus Nodestatus
profibus_read_byte.vi	Byte vom Profibus lesen
profibus_read_crate_controller.vi	Cratekontrollsystem auslesen
profibus_read_int.vi	Integer vom Profibus lesen
profibus_read_node_status.vi	ermittelt Status eines Profibus-Nodes
profibus_read_output_int.vi	Integer vom Ausgangsregister lesen
profibus_tagger_hv_chamber.vi	Abfrage der Taggerkammer
profibus_target.vi	Abfrage der Targetdaten
profibus_write_byte.vi	Byte zum Profibus schreiben
profibus_write_int.vi	Integer zum Profibus schreiben

Tabelle E.3: Vis für den Profibus-Zugriff

Dateiname	Funktion
beam_scan_read.vi	liest Beam-Scan Daten vom Dämon
beam_scan_request.vi	startet Beam-Scan
beam_scan_stop.vi	stoppt Beam-Scan
hall_probe_read.vi	liest Daten der Hall-Sonde vom Dämon
hv_off.vi	HV einschalten
hv_on.vi	HV ausschalten
hv_queue_status.vi	Status des Kommandopuffers abrufen
hv_read.vi	HV Daten vom Dämon lesen
slowcontroldaemon_check.vi	überprüft, ob Dämon läuft
slowctrld_transfer_data.vi	tranferiert Profibus-Daten zum Dämon

Tabelle E.4: Vis für Zugriff auf den Slowcontrol-Dämon

Dateiname	Funktion
hall_probe_check.vi	überprüft Daten der Hall-Sonde
hv_4032a.vi	Interface für LeCroy HV4032A HVs
hv_8_channels.vi	8 HV-Kanäle für HV-Interface
hv_check_all_changes_32_channels.vi	überprüft, ob neue Spannungen gesetzt (32 Kan.)
hv_check_all_changes_40_channels.vi	überprüft, ob neue Spannungen gesetzt (40 Kan.)
hv_check_all_channels.vi	überprüft Status aller Kanäle einer HV
hv_check_change_4channels.vi	überprüft 4 Kanäle auf neue Spannungen
hv_check_change.vi	überprüft auf neue Spannungen
hv_display_8_channels.vi	Anzeige von 8 HV-Kanälen
hv_gamma.vi	Interface für Gamma-Veto-HV
hv_get_status.vi	ermittelt den HV-Status
hv_scifi.vi	Interface für Innendetektor-HV
hv_set_channel.vi	Kanal auf neuen Spannungswert setzen
hv_set_tof.vi	TOF-HV-Kanal auf neuen Spannungswert setzen
hv_tagger.vi	Interface der Tagger-HV
hv_tof.vi	Interface der TOF-HV
hv_tof_check_status.vi	Status der TOF-HV prüfen (ein/aus)
hv_tof_combine_data.vi	TOF-HV-Daten zusammenfassen
hv_tof_get_status_from_voltages.vi	Status Kanäle durch Abweichung (Soll/Ist)
hv_tof_hv_mapping_to_tof_mapping.vi	Zuordnung HV-Kanäle → TOF Szintillatoren

Tabelle E.5: Vis zur Anzeige und Steuerung von Terminal-Server-Geräten

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] The CB-ELSA Collaboration, Study of baryon resonances decaying into $\Delta(1232)\pi^0$ in the reaction $\gamma\,\mathrm{p} \rightarrow \mathrm{p}\,\pi^0\,\pi^0$ with the Crystal Barrel detector at ELSA, Proposal to the PAC, 1998

[2] The CB-ELSA Collaboration, A search for the exotic meson $\hat{\rho}(1380)$ and for new baryonic resonances in the reaction $\gamma\,\mathrm{p} \rightarrow \mathrm{p}\,\pi^0\,\eta$ using the CB-ELSA detector at ELSA, Proposal to the PAC, 1999

[3] The CB-ELSA Collaboration, Inelastic Photon Scattering in the Exclusive Channels $\mathrm{p}(\gamma,\pi^0\gamma\mathrm{p})$ and $\mathrm{p}(\gamma,\eta\gamma\mathrm{p})$, Proposal to the PAC, 1999

[4] W. J. Schwille et al, Design and construction of the SAPHIR detector, Nuclear Instruments and Methods in Physics Research, A344 470-486, 1994

[5] Jürgen Wißkirchen, Entwicklung der Software und Aufbau der Szintillationszähler des Taggingsystems TOPASII, Diplomarbeit Physikalisches Institut, Universität Bonn, 1994

[6] Jürgen Link, Aufbau und Test der Proportionaldrahtkammern für das neue Taggingsystem an SAPHIR, Diplomarbeit Physikalisches Institut, Universität Bonn, 1994

[7] Bertram Kopf, Dissertation in Vorbereitung, Institut für Experimentalphysik I, Universität Bochum, 2001

[8] Angela Fösel, Entwicklung und Bau eines Innendetektors für das Crystal-Barrel-Experiment an ELSA/Bonn, Dissertation Physikalisches Institut, Universität Erlangen, 2000

[9] E. Aker et al., The Crystal Barrel spectrometer at LEAR, Nuclear Instruments and methods in Physics Research A321:69–108, 1992

[10] Andreas Ehmanns, Entwicklung, Aufbau und Test eines neuen Auslesesystems für den Crystal-Barrel-Detektor zur Messung photoinduzierter Reaktionen an ELSA, Dissertation Institut für Strahlen- und Kernphysik, Universität Bonn, 2000

[11] D. Jakob, Entwicklung eines Flugzeitspektrometers zur Messung der Quadrupolamplitude des $\mathrm{N} \rightarrow \Delta$ - Überganges, Dissertation Physikalisches Institut, Universität Bonn, 1996

[12] S. Höffgen, Einbindung eines großflächigen Flugzeitspektrometers als Vorwärtsdetektor für Experimente mit CB-ELSA, Diplomarbeit Physikalisches Institut, Universität Bonn, 1999

[13] The BaF_2 Photon Spectrometer TAPS, TAPS Collaboration (R. Novotny et al) Published in IEEE Trans. Nucl. Sci. Vol. 38 (1991) 379

[14] R. Tauchen, Test von Bleiglasdetektoren zum Elektronennachweis am ELAN Experiment, Diplomarbeit Physikalisches Institut, Universität Bonn, 1990

[15] R. Bantes, Dissertation in Vorbereitung, Physikalisches Institut, Universität Bonn, 2001

[16] M. Konrad, Ortssensitiver Detektor für hochenergetische Photonen bei höchsten Raten, Diplomarbeit Physikalisches Institut, Universität Bonn, 2001

[17] C. Schmidt, Optimierung des Datenakquisitions-Systems des Crystal Barrel Experiments an ELSA, Diplomarbeit Institut für Strahlen- und Kernphysik, Universität Bonn, 1999

[18] M. Fuchs, Entwicklung einer Run-Control zur Steuerung des Crystal-Barrel-Experiments an ELSA, Diplomarbeit Institut für Strahlen- und Kernphysik, Universität Bonn, 1999

[19] Horowitz und Hill, The art of electronics (second edition), Cambridge University Press, 1980/1989

[20] H. Schildt, C/C++: The complete reference (third edition), Osborne Publishing, 1998

[21] Siemens/Infineon Datenblatt, Plastic Fiber Optic Transmitter Diode and plastic connector housing

Danksagung

Zum Abschluss möchte ich allen danken, die zur Entstehung dieser Diplomarbeit beigetragen haben.

Ganz besonders möchte ich mich bei Herrn Prof. Dr. E. Klempt bedanken, der mir diese interessante Aufgabe anvertraut hat. Herrn Priv. Doz. Dr. R. W. Gothe danke ich für die Übernahme des Korreferats.

Herrn Dr. Hartmut Kalinowsky danke ich für seine geduldige Unterstützung bei allen auftretenden Fragen und seine Hilfe bei vielen Elektronikproblemen.

Großer Dank gilt auch der Arbeitsgruppe und der gesamten Crystal-Barrel-Kollaboration. Insbesondere bei meinen Zimmerkollegen Andreas Ehmanns, Michael Fuchs und Christoph Schmidt möchte ich mich für die fruchtbare Zusammenarbeit und die angenehme Arbeitsatmosphäre bedanken.

Und zum Schluss möchte ich mich auch bei meinen Eltern und ganz besonders bei meiner Freundin Kathrin für das Verständnis und die Unterstützung während des Studiums und der Diplomarbeit bedanken.